Erkki Mäkinen (toim.)

Tietojenkäsittelytieteellisiä tutkielmia
Syksy 2011

INFORMAATIOTIETEIDEN YKSIKKÖ
TAMPEREEN YLIOPISTO

INFORMAATIOTIETEIDEN YKSIKÖN RAPORTTEJA 5/2012

TAMPERE 2012

TAMPEREEN YLIOPISTO
INFORMAATIOTIETEIDEN YKSIKKÖ
INFORMAATIOTIETEIDEN YKSIKÖN RAPORTTEJA 5/2012
TAMMIKUU 2012

Erkki Mäkinen (toim.)

Tietojenkäsittelytieteellisiä tutkielmia
Syksy 2011

INFORMAATIOTIETEIDEN YKSIKKÖ
33014 TAMPEREEN YLIOPISTO

ISBN 978-951-44-8710-1

ISSN-L 1799-8158
ISSN 1799-8158

Aluksi

Tähän julkaisuun on kerätty syyslukukaudella 2011 pidetyn Tutkielmakurssin tutkielmia. Tutkielmien ohjaajina ovat lisäkseni toimineet Tomi Heimonen, Pentti Hietala, Kati Iltanen, Erno Mäkinen, Timo Poranen ja Sari Walldén.

Toimittaja

Sisällysluettelo

Vaatimusten määrittely kustannustehokkuuden näkökulmasta

Perttu Hallikainen

Tiivistelmä.
Vaatimusten määrittely on tärkeä osa ohjelmistotuotantoprojektia. Tämä prosessi määrää usein sen, kuinka hyvin projekti ja sen lopputulos tulevat onnistumaan. Perinpohjainen vaatimusten määrittely on kuitenkin aikaa vievää ja kallista. Tässä tutkielmassa perehdytään siihen, kuinka vaatimusten määrittelyn toteuttaminen voisi olla mahdollisimman kustannustehokasta.

Avainsanat ja -sanonnat: Vaatimusten määrittely, ohjelmistotuotanto.
CR-luokat: D.2.1

1. Johdanto

Ohjelmistotuotantoprojekti on hyvin laaja kokonaisuus ja yksi sen tärkeimmistä vaiheista on vaatimusten määrittely. Vaatimusten määrittelyn avulla pystytään selvittämään, mitkä ovat projektin päämäärät asiakkaan ja kehittäjien osalta [Sommerville, 2007]. Toisin sanoen määritellään, kuinka toteutettavan sovelluksen pitäisi toimia ja minkälaisia palveluita sen on tarkoitus tuottaa. Jotta projekti olisi onnistunut, on tärkeää pystyä selvittämään ja ymmärtämään nämä päämäärät riittävällä tarkkuudella. Sovellus, joka ei toteuta tarvittavia vaatimuksia, on hyödytön.

Vaatimusten määrittely on vaikea ja aikaa vievä prosessi ja prosessin kustannukset voivat nousta hyvin korkeiksi. Vaatimusten määrittely on tärkeää tehdä riittävällä tarkkuudella, sillä huonosti suunniteltu ja toteutettu sovellus aiheuttaa kustannuksia järjestelmän toimittamisen jälkeen. Hyvin toteutettu vaatimusten määrittely tulee huomattavasti edullisemmaksi kuin muutosten tekeminen jälkikäteen.

Ohjelmistotuotantoprojektin kokonaiskustannukset ovat usein sidoksissa projektiin kuluvaan aikaan. Aikatauluvaatimukset projekteille ovat yleensä hyvin tiukat, koska tämän päivän talousmalli ja markkinat vaativat sitä. Vaatimusten määrittelyyn käytettävät menetelmät riippuvat hyvin paljon kehitettävän järjestelmän luonteesta. On esimerkiksi selvää, että mitä kriittisempi järjestelmä on kyseessä, sitä enemmän tähän prosessiin joudutaan käyttämään aikaa.

Tässä tutkielmassa perehdytään vaatimusten määrittelyyn kustannustehokkuuden näkökulmasta. Aluksi luvussa 2 esitellään erilaiset ohjelmistoprojektiin liittyvät vaatimukset, kuten käyttäjävaatimukset ja järjestelmävaatimukset. Luvussa 3 esitellään yleisesti vaatimusmäärittelyprosessin vaiheet sekä kustannuksiin vaikuttavat tekijät.

Tämän jälkeen luvussa 4 perehdytään vaatimusten hallintaan ja sen merkitykseen projektin kustannuksissa. Vaatimusten hallinta on hankala ja kallis prosessi varsinkin suurten järjestelmien kohdalla. Näin ollen vaatimusten hallinta on tärkeä osa koko ohjelmistoprojektia.

Lopuksi luvuissa 5 ja 6 perehdytään menetelmiin, joilla kustannustehokkuutta voidaan parantaa erilaisissa ohjelmistoprojekteissa. Luvussa 5 esiteltävät muodolliset menetelmät ovat yksi tapa varmaan ja yksiselitteiseen vaatimusten määrittelyyn. Tällaista määrittelyä vaaditaan varsinkin kriittisissä järjestelmissä. Muodolliset menetelmät ovat kuitenkin hidas prosessi, joten ne eivät ole optimaalinen ratkaisu kaikkiin projekteihin.

Tämän päivän globaali kilpailu ja muuttuvat markkinat vaativat usein nopeaa ohjelmistotuotantoa. Luvussa 6 perehdytään nopeisiin menetelmiin, jotka sopivat hyvin pieniin ja keskikokoisiin järjestelmiin. Nämä järjestelmät eivät yleensä ole kriittisiä.

2. Vaatimukset

Vaatimukset ovat ohjelmistotuotantoprojektin alkuvaiheessa tehtäviä määrityksiä, jotka kertovat mitä tullaan toteuttamaan [Kotonya and Sommerville, 2004]. Ne ovat kuvauksia siitä, mitä sovellus tekee, miten sovellus tulee käyttäytymään, millaisessa ympäristössä sovellus toimii ja millaisia rajoitteita sovelluksella on.

Vaatimukset voidaan luokitella kahteen eri pääluokkaan. Nämä luokat ovat käyttäjävaatimukset ja järjestelmävaatimukset [Sommerville, 2007]. Käyttäjävaatimukset ovat yleisiä luonnollisella kielellä tehtyjä toteamuksia, jotka määrittävät, mitä palveluita järjestelmän pitäisi tuottaa ja millaisten rajoitusten mukaisesti. [Sommerville, 2007]

Järjestelmävaatimukset taas määrittelevät yksityiskohtaisesti järjestelmän toiminnallisuudet, palvelut ja rajoitteet. Järjestelmävaatimusten on tarkoitus määritellä yksiselitteisesti se, mitä tullaan toteuttamaan. [Sommerville, 2007]

Järjestelmä- ja käyttäjävaatimukset voidaan jakaa kolmeen alempaan luokkaan. Nämä luokat ovat funktionaaliset ja ei-funktionaaliset vaatimukset sekä ympäristövaatimukset.

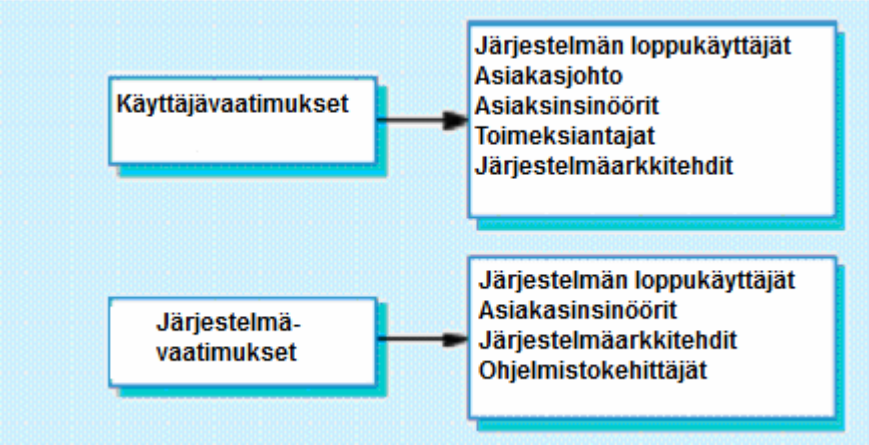
Funktionaaliset vaatimukset ovat toteamuksia, jotka kertovat, millaisia palveluita järjestelmän pitäisi tuottaa. Nämä vaatimukset kertovat myös, kuinka järjestelmän pitäisi reagoida syötteisiin ja kuinka sen pitäisi käyttäytyä eri tilanteissa. [Sommerville, 2007]

Ei-funktionaaliset vaatimukset ovat vaatimuksia, jotka eivät varsinaisesti keskity järjestelmän toiminnallisuuteen. [Kotonya and Sommerville, 2004]. Nämä vaatimukset ovat enemmänkin rajoitteita, jotka liittyvät projektiin ja järjes-

telmään. Ei-funktionaaliset vaatimukset sisältävät muun muassa turvallisuusvaatimukset, varmuusvaatimukset, käytettävyysvaatimukset, luotettavuusvaatimukset ja suorituskykyvaatimukset [Kotonya and Sommerville, 2004].

Ympäristövaatimukset ovat vaatimuksia, jotka tulevat sovelluksen ympäristöstä. Nämä ovat rajoitteita, joita järjestelmän toimintaympäristö aiheuttaa järjestelmälle. [Sommerville, 2007]

Eri tarkkuudella tehtävät vaatimusmäärittelyt ovat erittäin tärkeitä, koska erilaiset ihmiset ovat tekemisissä vaatimusten kanssa. Sovelluksen kehittäjät ja asiakas eivät esimerkiksi välttämättä ymmärrä vaatimuksia samalla tavalla. Luonnollisen kielen yleinen kuvaus ei ole riittävän yksityiskohtainen, jotta kehittäjä ymmärtäisi, mitä pitää toteuttaa. Asiakas taas ei välttämättä ymmärrä tarkkoja tietoteknisiä järjestelmävaatimuksia, koska hänellä ei ole alan asiantuntemusta. Eri käyttäjiä järjestelmä- ja käyttäjävaatimuksille on kuvattu kuvassa 1.



Kuva 1. Käyttäjä- ja järjestelmävaatimukset. [Sommerville, 2007]

Epätarkkuus järjestelmävaatimusten määrittelyssä on yksi suurimmista ongelman aiheuttajista ohjelmistotuotantoprojektissa [Sommerville, 2007]. Jos vaatimusten määrittely ei ole yksiselitteistä, on melko yleistä, että ohjelmistokehittäjät pyrkivät toteuttamaan sovelluksen omasta näkökulmastaan. Tämä aiheuttaa usein sen, että järjestelmän toteutus ei vastaa asiakkaan alkuperäistä päämäärää. Tällaisessa tapauksessa joudutaan usein tekemään muutoksia jälkikäteen sekä vaatimuksiin että itse järjestelmään. Tämä viivästyttää järjestelmän valmistumista ja lisää projektin kustannuksia.

2.1. Funktionaaliset vaatimukset

Funktionaalisten vaatimusten määrittelemisessä pitäisi pyrkiä mahdollisimman aukottomaan ja yhdenmukaiseen lopputulokseen [Sommerville, 2007]. Tämä tarkoittaa sitä, että kaikki vaadittavat käyttötarkoitukset ja palvelut tulevat esille asiakkaan puolelta. Näiden vaatimusten pitää olla myös täysin yksiselitteisiä ja helposti ymmärrettäviä.

Yksiselitteisten vaatimusten muodostaminen voi olla joissakin tapauksissa hyvin vaikeaa. Varsinkin suuressa projektissa vaatimuksiin jää helposti aukkoja ja ristiriidat ovat todennäköisiä. Suuri syy ristiriidoille ovat lukuisat käyttäjät. Järjestelmällä saattaa olla satoja eri käyttäjiä, jotka käyttävät järjestelmää eri tarkoituksiin. Eri käyttäjien vaatimukset voivat olla ristiriidassa keskenään, mikä aiheuttaa ongelmia sovelluksen toiminnallisuudessa. Pahimmillaan nämä ristiriidat huomataan vasta, kun järjestelmä on jo toimitettu asiakkaalle. Suuren järjestelmän muokkaaminen toimituksen jälkeen voi tulla asiakkaalle hyvin kalliiksi.

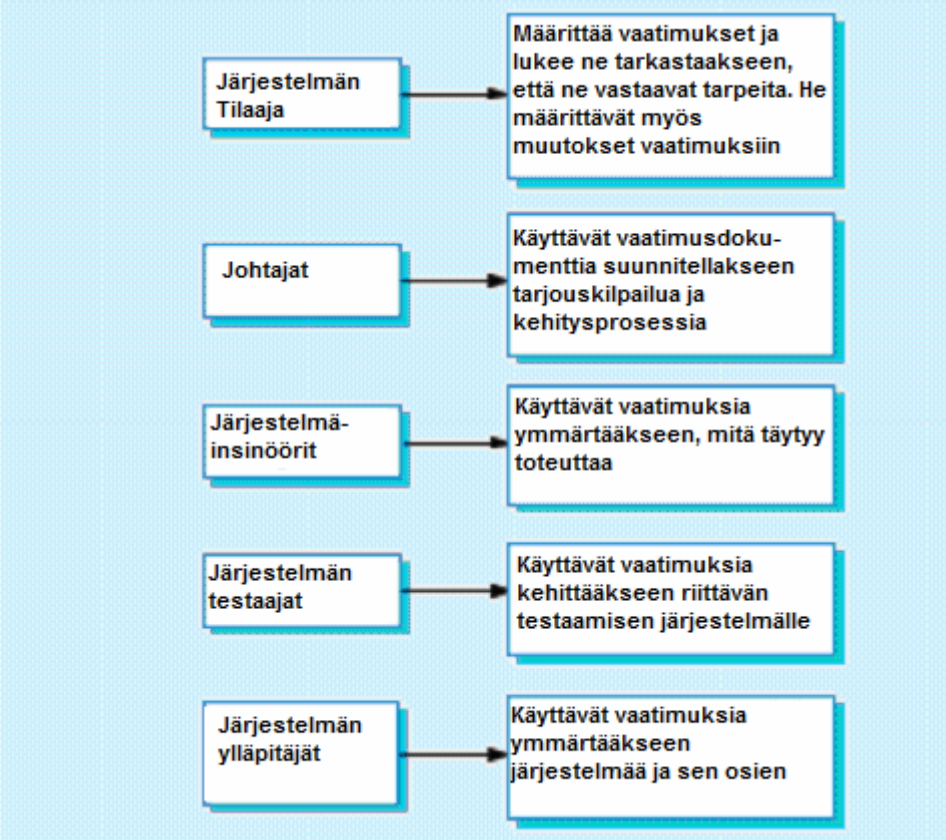
2.2. Ei-funktionaaliset vaatimukset

Ei-funktionaalisten vaatimusten määrittelemisessä yleisin ongelma on, että näitä vaatimuksia on vaikea todentaa ja mitata [Sommerville, 2007]. Asiakkaalta saatetaan esimerkiksi saada vaatimus, että järjestelmän täytyy olla helppokäyttöinen ja turvallinen. Nämä vaatimukset ovat hyvin epätarkkoja ja vaikeasti mitattavissa. Epätarkkuus voi pahimmillaan aiheuttaa sen, että järjestelmä ei vastaa asiakkaan oikeita tavoitteita ja päämääriä.

Ei-funktionaaliset vaatimukset pitäisi aina pyrkiä määrittelemään kvantitatiivisessa muodossa niin, että vaatimuksia voidaan mitata ja testata [Kotonya and Sommerville, 2007]. Kun nämä kaksi vaatimusta täyttyvät, voidaan järjestelmää testattaessa yksiselitteisesti varmistua, että järjestelmä vastaa vaatimuksia.

Asiakkaalle voi olla vaikeaa tai lähes mahdotonta määritellä ei-funktionaalisia vaatimuksia kvantitatiivisessa muodossa. Asiakkaan asiantuntemuksen puute on usein ongelma. Jos asiakas ei omaa hyvää tietotekniikan tuntemusta, voi hänen esimerkiksi olla vaikeaa tehdä vaatimuksia suorituskyvyn tai muistinkäytön osalta. Täydellinen vaatimusten tekeminen mitattavaan muotoon voi myös tulla hyvin kalliiksi, joten asiakas ei välttämättä ole halukas tekemään tarkkaa määrittelyä.

Vaatimusten määrittelyn lopputuloksena on tarkoitus syntyä vaatimusdokumentti. Dokumentin eri käyttäjät on kuvattu kuvassa 2. Dokumentti on virallinen lausunto siitä, mitä järjestelmän kehittäjien pitää toteuttaa [Sommerville, 2007]. Sen on tarkoitus palvella jokaista osapuolta, kuten asiakasta, järjestelmän kehittäjiä, projektin johtajia sekä käyttäjiä [Kotonya and Sommerville, 2004]. Dokumentin tarkkuus riippuu toteutettavasta järjestelmästä ja projektissa käytettävistä menetelmistä. Kriittiset järjestelmät vaativat erittäin tarkkaa ja laajaalaista vaatimusten määrittelyä. Kriittiset järjestelmät ovat järjestelmiä, joiden toimintahäiriöstä voi koitua todella suuria kustannuksia tai muita haittavaikutuksia.



Kuva 2. Vaatimusdokumentin käyttäjät. [Sommerville, 2007]

3. Vaatimusten määritteleminen

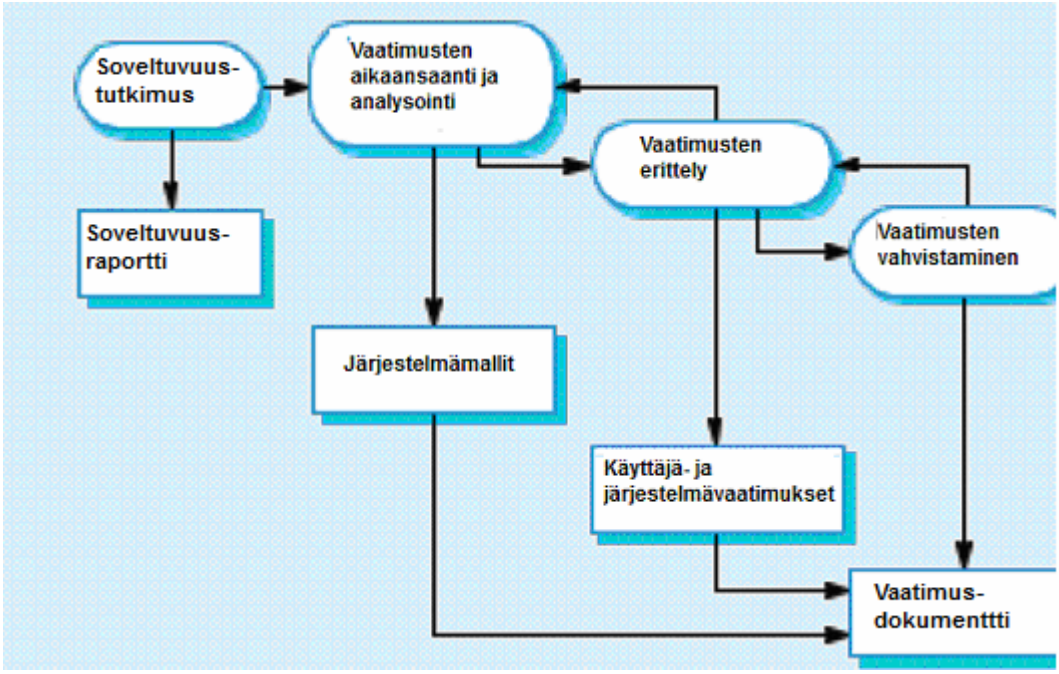
Vaatimusten määrittely on prosessi, jonka avulla vaatimusdokumentti tehdään. Tämä prosessi auttaa sovelluksen kehittäjiä ymmärtämään, mitä heidän pitää toteuttaa ja millaisia ongelmia heidän pitää ratkaista.

Prosessi sisältää erilaisia tehtäviä, joiden avulla saavutetaan ymmärrys projektin luonteesta. Prosessin on tarkoitus vastata seuraaviin kysymyksiin: Mikä tulee olemaan järjestelmän todellinen käyttötarkoitus? Mitä asiakas haluaa? Kuinka järjestelmän käyttäjät lopulta käyttävät järjestelmää? [Pressman, 2005]

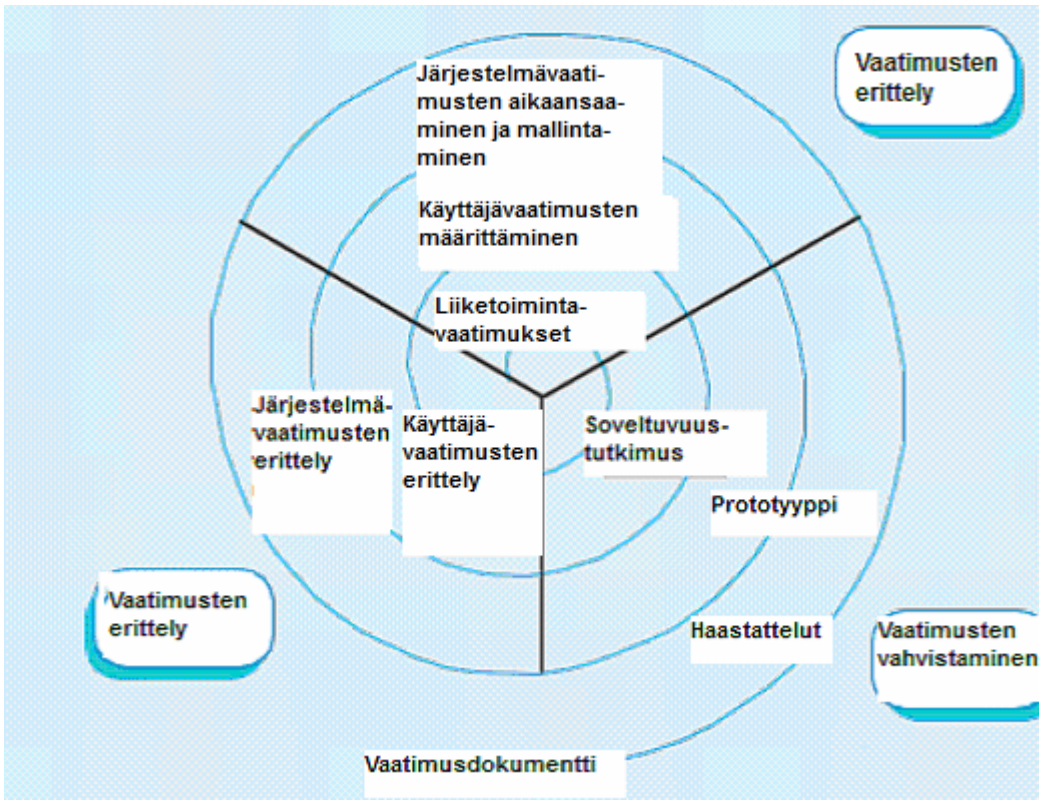
Vaatimusmäärittelyprosessin voidaan katsoa sisältävän neljä vaihetta. Nämä vaiheet ovat soveltuvuustutkimus, vaatimusten aikaansaanti ja analysointi, vaatimusten erittely ja vaatimusten vahvistaminen [Sommerville, 2007]. Nämä vaiheet sisältyvät pääprosessiin, jota sanotaan vaatimusten kehittämiseksi. Toinen pääprosessi on vaatimusten hallinnointi. Vaatimusten hallinnointi sisältää muun muassa kaikki toiminnot, jotka liittyvät vaatimusten muuttamiseen ja lisäämiseen [Attarha and Modiri, 2011].

Vaatimusmäärittelyprosessissa voidaan käyttää erilaisia malleja. Prosessissa voidaan käyttää esimerkiksi vesiputousmallia (kuva 3) tai kierremallia (kuva 4). Vesiputousmallissa prosessi etenee vaiheittain. Tämä tarkoittaa sitä, että seu-

raavaan prosessin vaiheeseen siirrytään vasta, kun edellinen vaihe on valmis. Tarvittaessa prosessissa voidaan palata taaksepäin.



Kuva 3. Vesiputousmalli. [Sommerville, 2007]



Kuva 4. Kierremalli. [Sommerville, 2007]

Kierremalli on iteratiivinen prosessi, jossa samat vaiheet käydään läpi useaan otteeseen, kunnes tarvittavat vaatimukset on onnistuttu toteuttamaan riittävällä tarkkuudella. Alkuvaiheen iteraatioissa pyritään keskittymään yleisellä tasolla ei-funktionaalisiin vaatimuksiin ja liiketoiminnan edellyttämiin vaatimuksiin [Sommerville, 2007]. Myöhemmissä iteraatioissa keskitytään enemmän järjestelmävaatimuksiin ja järjestelmän mallintamiseen.

3.1. Soveltuvuustutkimus

Kaikkien uusien järjestelmien kohdalla vaatimusmäärittelyprosessin pitäisi alkaa soveltuvuustutkimuksella [Sommerville, 2007]. Tässä tutkimuksessa selvitetään, kuinka järjestelmän on tarkoitus tukea liiketoimintaa tai tarkoitusta, johon järjestelmä tulee. Toisin sanoen on tarkoitus selvittää, mitä asiakas hyötyy järjestelmästä. Tutkimuksen lopputulos kertoo, kannattaako projektia jatkaa.

Joissakin tapauksissa soveltuvuustutkimus ei ole pakollinen ja pelkkä keskustelu asiakkaan kanssa riittää [Pressman, 2005]. Suurimmassa osassa projekteja tämä vaihe on kuitenkin välttämätön. Varsinkin, jos kyseessä on kokonaan uusi idea tai jos järjestelmä on tulossa kokonaan uuteen tarkoitukseen, on syytä toteuttaa tämä tutkimus huolella.

Soveltuvuustutkimuksen on tarkoitus vastata ainakin seuraaviin kysymyksiin: Tukeeko järjestelmä organisaation yleisiä tavoitteita? Voidaanko järjestelmä toteuttaa käyttäen nykyistä teknologiaa? Voidaanko järjestelmä toteuttaa annetuissa kustannus- ja aikarajoissa? Voidaanko järjestelmä integroida nykyisin käytettäviin välttämättömiin järjestelmiin? [Sommerville, 2007]

Soveltuvuustutkimus voidaan toteuttaa haastattelemalla eri alan asiantuntijoita. Tutkimuksen tekijän kannattaa pyrkiä konsultoimaan ohjelmistokehittäjiä, jotka ovat työskennelleet samankaltaisen projektin parissa ja jotka tuntevat hyvin käytettävää teknologiaa [Pressman, 2005]. On tärkeää haastatella myös asiakasyrityksen johtoa sekä järjestelmän lopullisia käyttäjiä.

Soveltuvuustutkimus kannattaa toteuttaa huolellisesti, jotta yritys ei tilaa itselleen järjestelmää, joka on hyödytön. Järjestelmä, joka ei palvele asiakkaan tarkoitusperiä tai päämääriä, tulee asiakkaalle hyvin kalliiksi. Yrityksen johdon ja asiakkaan on tärkeää ymmärtää, mitkä ovat heidän vaatimukset liiketoiminnan edistämisen suhteen ja kuinka järjestelmän pitäisi edistää heidän toimintaansa.

3.2. Vaatimusten aikaansaanti ja analysointi

Vaatimusten aikaansaanti ja analysointi on vaihe, jossa ohjelmistokehittäjät työskentelevät yhdessä asiakkaan kanssa. Tavoitteena on saada vastaukset ainakin seuraaviin kysymyksiin: Mitä palveluita järjestelmän pitäisi tuottaa? Mi-

ten järjestelmän kuuluisi toimia? Millaisessa sovellusympäristössä järjestelmä tulee toimimaan? [Sommerville, 2007]

Vaatimusten aikaansaanti ei ole pelkästään sitä, että asiakas kertoo, mitä hän järjestelmältä haluaa. Tämä prosessi vaatii tarkkaa perehtymistä organisaatioon, sovellusympäristöön ja liiketoimintamalliin, johon järjestelmä on tarkoitettu [Kotonya and Sommerville, 2004].

Asiakkaan vaatimusten ymmärtäminen voi olla ongelmallista ohjelmistokehittäjille useista syistä. On mahdollista, että asiakas ei tiedä, mitä haluaa järjestelmältä, tai ei osaa ilmaista asiaa oikein. On myös mahdollista, että asiakas tekee täysin epärealistisia vaatimuksia ymmärtämättä niiden vaikutuksia kustannuksiin tai aikatauluun. Usein tulee myös tilanteita, jolloin kehittäjät ja asiakas eivät ymmärrä toisiaan tai ymmärtävät vaatimukset väärin.

Iso ongelma vaatimusten määrittelyssä ovat epävakaat vaatimukset. Epävakaat vaatimukset tarkoittavat vaatimuksia, jotka tulevat todennäköisesti muuttumaan projektin myöhemmissä vaiheissa. [Pressman, 2005]. On tärkeää pyrkiä jo projektin alkuvaiheessa määrittämään, mitkä vaatimukset ovat pysyviä ja mitkä epävakaita.

Vaatimusten analysointi ja neuvottelu on prosessi, joka kuuluu vaatimusten aikaansaantiin [Kotonya and Sommerville, 2004]. Tämä on prosessi, jonka tavoitteena on yksiselitteiset vaatimukset. Analysointiprosessin aikana on tarkoitus löytää päällekkäiset vaatimukset, ristiriidassa olevat vaatimukset, moniselitteiset vaatimukset, puuttuvat vaatimukset sekä epärealistiset vaatimukset. Mikäli päällekkäisiä tai ristiriidassa olevia vaatimuksia löytyy, on asiakkaan kanssa neuvoteltava kompromisseista ja vaatimusten yksinkertaistamisesta.

Vaatimuksia on syytä analysoida, jotta voidaan selvittää, miten vaatimus oikeasti toimii järjestelmässä. Tämä onnistuu mallintamalla toimintaa esimerkiksi käyttäjäskenaarioiden avulla. Käyttäjäskenaarioilla kuvataan tässä tapauksessa sitä, kuinka loppukäyttäjä oikeasti tulee käyttämään järjestelmää ja kuinka vaatimukset vastaavat näitä käyttötarkoituksia [Pressman, 2005].

Vaatimusten aikaansaamisessa ja analysoinnissa käytetään yleensä kierremallia, joka esitettiin kuvassa 4. Tässä kierremalli toimii omana vaiheenaan, ja kierre sisältää neljä toimintoa. Toiminnot ovat vaatimusten löytäminen, vaatimusten luokitteleminen, vaatimusten priorisoiminen ja vaatimusten dokumentoiminen [Sommerville, 2007].

3.3. Vaatimusten erittely

Ohjelmistokehityksen yhteydessä vaatimusten erittely voi tarkoittaa eri asioita eri tilanteissa. Erittely voi olla esimerkiksi dokumentti, piirretty malli, tietty määrä skenaarioita, prototyyppi tai yhdistelmä näitä [Pressman, 2005]. Erittelyn on tarkoitus kuvailla järjestelmän todellista toiminnallisuutta sekä rajoituksia.

Ei ole tarkkaan määriteltyä, millä tavalla tämän vaiheen lopputulos tulisi esittää, mutta sen merkitys on selvä. Vaatimusten erittelyn on tarkoitus selkeyttää vaatimuksia ja tehdä niistä ymmärrettävämpiä. Esimerkiksi laajaa monimutkaista järjestelmää voi olla hankala kuvata pelkästään yhdellä tavalla. Hyvä keino erittelyyn voisi olla yhdistelmä, joka koostuu kirjoitetusta dokumentista, luonnollisesta kielestä ja piirretyistä malleista [Pressman, 2005]. Pienemmissä projekteissa, joiden tekniikka tunnetaan hyvin, riittää usein pelkästään käyttäjäskenaariot.

3.4. Vaatimusten vahvistaminen

Vaatimusten vahvistaminen on tärkeimpiä vaiheita vaatimusten määrittelyä tehtäessä. Tässä vaiheessa tarkastetaan, että vaatimukset ovat kattavia, yksiselitteisiä, hyvin kirjoitettuja ja että ne vastaavat asiakkaan tarpeita [Attarha and Modiri, 2005]. Tässä vaiheessa keskitytään myös löytämään vaatimuksiin liittyvät ongelmat.

Vahvistaminen on tärkeää suorittaa huolellisesti, jotta virheiltä ja ristiriidoilta vältyttäisiin. Ristiriidat ja virheet aiheuttavat muutoksia jälkikäteen ja nämä muutokset taas aiheuttavat lisätyötä ja kustannuksia. Kustannukset nousevat varsinkin, mikäli ongelmakohdat huomataan vasta, kun järjestelmä on toimitettu asiakkaalle. Vaatimusmäärittelyvirheiden paikkaaminen jälkikäteen tulee huomattavasti kalliimmaksi kuin esimerkiksi suunnitteluvirheet tai ohjelmointivirheet [Sommerville, 2007]. Tämä johtuu siitä, että kun vaatimuksia täytyy muuttaa, täytyy yleensä myös muuttaa järjestelmän suunnittelua ja järjestelmän toteutusta. Tämä tarkoittaa myös sitä, että järjestelmää pitää testata uudelleen.

Vaatimusten vahvistamisessa täytyy tarkastaa, että vaatimukset ovat tarpeellisia, johdonmukaisia, realistisia ja ymmärrettäviä. Kustannuksien kannalta on tärkeää arvioida, onko vaatimus ylipäänsä mahdollista toteuttaa ja onnistuuko se annetun budjetin ja aikataulun puolesta.

Pääasiallinen tapa vaatimusten vahvistamisen tekemiseen on tekninen haastattelu, joka toteutetaan kaavamaisesti samalla tavalla kaikille [Pressman, 2005]. Haastattelussa pyritään ottamaan huomioon kaikki, jotka ovat tekemisissä järjestelmän kanssa. Jokainen osapuoli käy läpi vaatimukset ja haastattelujen avulla on tarkoitus löytää mahdolliset ristiriidat, ongelmat, epäselvät vaatimukset, puuttuvat vaatimukset sekä epäjohdonmukaisuudet [Pressman, 2005].

Haastatteluiden lisäksi erittäin tehokas tapa vaatimusten vahvistamiseen on prototyypin käyttö [Kotonya and Sommerville, 2004]. Vaatimusten pohjalta rakennetun prototyypin avulla järjestelmän toiminnallisuutta on helppo demonstroida ja testata. Tämän avulla virheet ja ristiriidat on helpompi löytää.

Lopuksi vaatimukset dokumentoidaan siten, että käyttäjät ja kehittäjät ymmärtävät vaatimukset ja niiden päämäärät yksiselitteisesti. Hyvä dokumentti

on virheetön, muokattava, kehitettävä ja ymmärrettävä [Attarha and Modiri, 2011]. Dokumentin pitää olla selkeä ja hyvin luokiteltu, jotta vaatimuksia on helppo hallita niiden muuttuessa. Lisäysten tekeminen dokumenttiin on onnistuttava vaivattomasti, koska on hyvin todennäköistä, että vaatimuksia tulee lisää myöhemmissä projektin vaiheissa.

4. Vaatimusten hallinnointi

Suurten järjestelmien vaatimuksissa tapahtuu usein muutoksia. Syitä muutoksille löytyy paljon. Muutosten tarve voi johtua asiakkaan tarpeista, muutoksista sovellusympäristössä, muutoksista liiketoiminnassa tai lakimuutoksista [Kotonya and Sommerville, 2004]. Vaatimukset voivat muuttua myös sen takia, että alkuperäiset vaatimukset ovat olleet puutteellisia tai epäselviä.

Kun valmis järjestelmä toimitetaan asiakkaalle, on hyvin yleistä, että vaatimuksia ilmestyy lisää. Tämä johtuu siitä, että asiakkaan on hyvin vaikea ennakoida sitä, kuinka uusi järjestelmä todellisuudessa tulee vaikuttamaan organisaatioon ja sen toimintaan. Yksi merkittävä syy sille, miksi loppukäyttäjät eivät ole tyytyväisiä järjestelmään, on se, että järjestelmän maksaja on yleensä eri henkilö kuin järjestelmän varsinainen käyttäjä [Sommerville, 2007]. Tämä johtaa usein huonoon lopputulokseen, koska käyttäjien vaatimuksissa tingitään kustannus- ja aikataulusyistä.

Vaatimusten hallinnointi on prosessi, jolla kontrolloidaan vaatimusten muutoksia [Kotonya and Sommerville, 2004]. Vaatimusten hallinnointi alkaa heti projektin alkuvaiheessa ja kestää koko järjestelmän elinkaaren ajan [Attarha and Modiri, 2011]. Vaatimukset täytyy ymmärtää hyvin ja niiden väliset riippuvuudet täytyy olla selvillä, jotta pystytään ennakoimaan muutosten vaikutuksia muihin vaatimuksiin. Muutosten tekemiseen on syytä käyttää kaavamaista prosessia, jonka avulla tulee otettua huomioon kaikki toisistaan riippuvaiset vaatimukset [Sommerville, 2007].

Vaatimusten hallinnoinnin kannalta vaatimukset kannattaa jakaa kahteen eri luokkaan: kestäviin vaatimuksiin ja epävakaisiin vaatimuksiin [Sommerville, 2007]. Kestävät vaatimukset ovat sellaisia, joista tiedetään, että ne tulevat pysymään sellaisinaan. Epävakaat vaatimukset ovat taas sellaisia, jotka voivat hyvin todennäköisesti tulla muuttumaan. Jakamalla nämä vaatimukset voidaan mahdollisesti ennakoida muutoksien aiheuttamia vaikutuksia ja vaatimusten välisiä riippuvuussuhteita.

Vaatimusten hallinnointi on yleensä kallis prosessi, joten se kannattaa suunnitella hyvin etukäteen. Kaikki vaatimukset tulisi yksilöidä, jotta jokainen yksittäinen vaatimus olisi helposti jäljitettävissä. On tärkeää myös selvittää kaikkien vaatimusten väliset riippuvuussuhteet toisiinsa. Tämän lisäksi pitäisi myös ke-

hittää yhdenmukainen muutoshallintaprosessi, jonka avulla pystytään arvioimaan muutosten aiheuttamia kustannuksia ja vaikutuksia muihin vaatimuksiin. Muutosten täytyy olla kontrolloituja, jotta voidaan varmistaa, että ne ovat kannattavia ja hallittuja [Kotonya and Sommerville, 2004]. On tärkeää myös tarkastella, että muutos on mahdollista tehdä käytössä olevan budjetin ja aikataulun rajoissa.

Vaatimusten välillä on paljon erilaisia riippuvuuksia, joten vaatimusten jäljitettävyys on tärkeää. Toisin sanoen täytyy olla helppo havaita, mihin tietty vaatimus vaikuttaa. Tätä informaatiota kuvataan yleensä jäljitettävyysmatriisien avulla [Sommerville, 2007]. Esimerkki jäljitettävyysmatriisista on esitetty kuvassa 5.

Req. id	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1		D	R					
1.2			D			D		D
1.3	R			R				
2.1			R		D			D
2.2								D
2.3		R		D				
3.1								R
3.2							R	

Kuva 5. Jäljitettävyysmatriisi. [Sommerville, 2007]

Kuvassa 5 kirjain D kuvaa sitä, että rivillä oleva vaatimus on riippuvainen samassa sarakkeessa olevasta vaatimuksesta. Kirjain R tarkoittaa sitä, että näiden kyseisten vaatimusten välillä on jokin heikompi riippuvuus.

Käytettävyysmatriisit ovat hyviä pienehköjen projektien kohdalla, kun vaatimuksia on vähän. Suurten projektien kohdalla jäljitettävyysmatriisit ovat kuitenkin hankalia hallita ja kalliita ylläpitää. Suurten projektien kohdalla on järkevämpää pitää vaatimustietokantaa, jossa jokainen yksittäinen vaatimus on linkitetty niihin vaatimuksiin, joiden välillä on riippuvuuksia [Sommerville, 2007]. Näin jäljitettävyysmatriisit voidaan luoda automaattisesti tietokantakyselyillä.

Käytettävä vaatimusmuutosprosessi pitäisi olla sovellettavissa kaikkiin vaatimuksiin. Prosessi täytyy olla kaavamainen, jotta kaikkia muutosehdotuksia käsiteltäisiin samalla tavalla ja että muutokset toteutetaan kontrolloidusti. Vaatimusmuutosprosessiin kuuluu kolme tasoa, jotka ovat ongelman arviointi ja muutoksen määrittäminen, muutoksen ja kustannusten arviointi ja muutoksen toteuttaminen [Sommerville, 2007].

Muutosprosessi alkaa sillä, että määritellään ongelma, jonka takia muutos halutaan tehdä. Tämän jälkeen arvioidaan, onko muutos tarpeellinen ja ratkaiseeko se ongelman. Sen jälkeen täytyy arvioida muutoksen aiheuttamia vaikutuksia muihin vaatimuksiin ja muutoksen aiheuttamia kustannuksia projektille. Kustannuksia arvioidaan mittaamalla vaikutuksia vaatimusdokumenttiin, suunnitteluun ja järjestelmän toteutukseen. Tämän jälkeen tehdään päätös siitä, toteutetaanko muutos. Mikäli muutos tehdään, on toteutettava tarvittavat muutokset vaatimusdokumenttiin, suunnitteluun ja järjestelmän toteutukseen. Jotta muutoksen kustannukset pysyisivät mahdollisimman pienenä, on tärkeää, että vaatimusdokumentti on mahdollisimman muokattava. Hyvä vaatimusdokumentti ei vaadi uudelleenorganisointia tai suuria muutosketjuja.

5. Muodolliset menetelmät

Termi muodollinen menetelmä tarkoittaa mitä tahansa matemaattista tapaa kuvata ohjelmistoa. Muodollinen määrittely tarkoittaa määrittelyä, joka on esitetty muodollista syntaksia ja semantiikkaa käyttäen. Toisin sanoen määrittely perustuu matemaattisiin malleihin, joiden ominaisuudet tunnetaan hyvin. [Sommerville, 2007]

On tutkittu, että vaatimuksista johtuvien virheiden korjaaminen voi olla jopa 10–100 kertaa kalliimpaa ohjelmistokehitys projektin myöhemmissä vaiheissa kuin vaatimusmäärittelyprosessin aikana [Hamilton et al., 1995]. Muodolliset menetelmät ovat erittäin hyvä tapa löytää virheet vaatimuksista ennen kuin ohjelmiston varsinainen toteutus alkaa. Muodollinen määrittely auttaa esittämään vaatimukset täysin yksiselitteisellä tavalla. Tämän menetelmän käyttäminen on tapa selvittää jo ennen ohjelmiston toteuttamista, että järjestelmä vastaa sille asetettuja vaatimuksia [Kotonya and Sommerville, 2004].

Vaikka muodolliset menetelmät ovat varma tapa tuottaa ja testata vaatimuksia, liittyy niihin kuitenkin omat ongelmansa. Muodollisten menetelmien käyttäminen on aikaa vievä ja hidas prosessi. Nykyajan markkinajärjestelmä vaatii järjestelmän mahdollisimman nopeaa toimittamista. Asiakkaalle on tärkeää, että järjestelmä saadaan nopeasti käyttöön. Asiakas on usein valmis jopa tinkimään järjestelmän laadusta, mikäli järjestelmä pystytään toimittamaan nopeammin.

Prosessin hitauden lisäksi muodollisilla menetelmillä on hankala kuvata käyttäjärajapintoja ja käyttäjän vuorovaikutusta. [Sommerville, 2007]. Menetelmä toimii myös heikosti erittäin laajojen järjestelmien kohdalla ja useissa yrityksissä ei ole riittävää osaamista muodollisten menetelmien toteuttamiseen [Kotonya and Sommerville, 2004]. Näistä syistä johtuen monet yritykset eivät ole halukkaita käyttämään muodollisia menetelmiä ohjelmistoprojekteissaan.

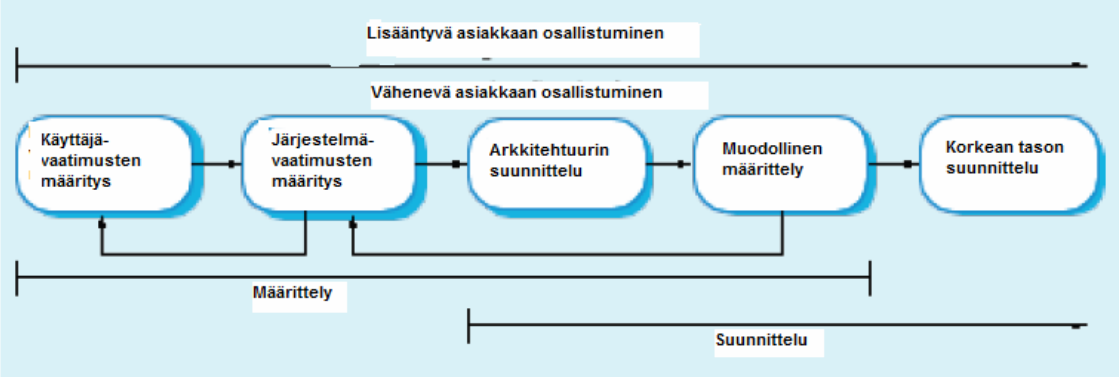
Näistä puutteista huolimatta matemaattisten mallien tuomista hyödyistä löytyy vahvaa näyttöä. Monet yritykset, jotka ovat käyttäneet muodollisia menetelmiä, ovat raportoineet hyvistä tuloksista [Sommerville, 2007]. Näistä raporteista käy ilmi, että virheiden määrä järjestelmässä on ollut hyvin pieni. Muodollisten menetelmien käyttö ei myöskään ole erityisesti kasvattanut projektin kokonaiskustannuksia.

Näiden raportointien perusteella näyttää siltä, että muodolliset mallit voivat olla kustannustehokas tapa, jos niiden käyttäminen rajoittuu pelkästään järjestelmän ydinosiin ja päätoimintoihin. Näin voidaan taata, että järjestelmän kriittisimmät osat toimivat varmasti ja näiden toimintojen virheiden määrä on minimoitu. Yritysten täytyy kuitenkin olla valmiita investoimaan teknologiaan, jota näiden menetelmien käyttäminen edellyttää.

Muodollisten mallien käyttäminen on yleistynyt paljon varsinkin kriittisten järjestelmien kehittämisessä [Sommerville, 2007]. Näissä järjestelmissä yleensä mittarit, kuten turvallisuus ja käyttövarmuus, ovat erittäin tärkeitä. Muodolliset menetelmät ovat varma tapa varmistaa, että järjestelmä vastaa asetettuja vaatimuksia. Tämä tietenkin edellyttää sitä, että vaatimukset on muotoiltu oikein kvantitatiiviseen muotoon.

Kriittisten järjestelmien toimintahäiriöt voivat aiheuttaa isoja kustannuksia tai jopa katastrofin, joten järjestelmän tilaaja on usein valmis investoimaan muodollisten menetelmien käyttöön. Voidaan sanoa, että muodollisia malleja käyttämällä pystytään ennaltaehkäisemään toimintavirheistä koituvia kustannuksia ja varmistaa järjestelmän käyttövarmuus sekä hyvä laatu.

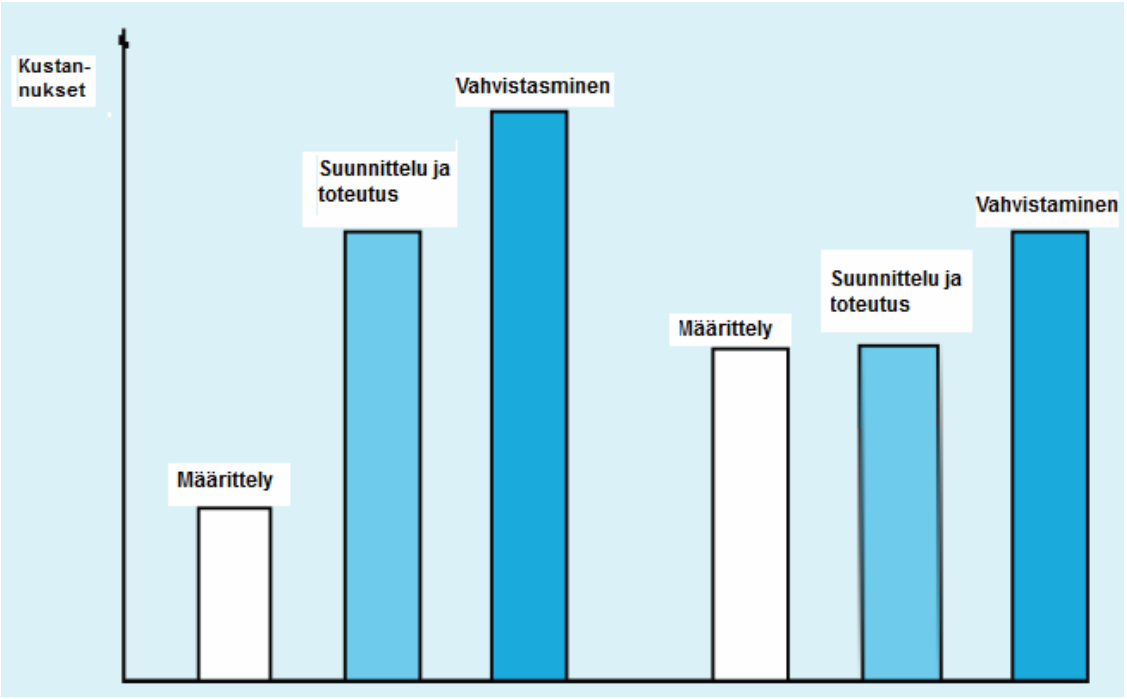
Muodollista määrittelyä käytettäessä järjestelmävaatimukset analysoidaan huolellisesti ennen kuin järjestelmän toteuttaminen aloitetaan. Jos tätä tapaa käytetään, tapahtuu se yleensä vaatimusten määrittelyn ja suunnittelun välissä [Sommerville, 2007]. Vaatimusten erittelyn ja muodollisen määrittelyn välillä on jatkuva silmukka, jonka avulla ongelmat ja epäselvyydet havaitaan. Tämä prosessi on esitetty kuvassa 6.



Kuva 6. Vaatimusten erittely ja muodollinen määrittely [Sommerville, 2007].

Muodollisten menetelmien käyttäminen lisää ymmärrystä ja tarkkuutta määrittelyihin [Sommerville, 2007]. Muodollisen erittelyn toteuttaminen pakottaa tekemään yksityiskohtaisen järjestelmäanalyysin, joka tuo esiin virheet ja epäjohdonmukaisuudet. Näiden virheiden ja epäjohdonmukaisuuksien korjaaminen voisi tulla hyvin kalliiksi myöhemmissä vaiheissa tai järjestelmän käyttöönoton jälkeen.

Muodollisten menetelmien vaikutuksia projektin kokonaiskustannuksiin ja virheiden määrään on empiirisesti tutkittu. On todistettu, että mitä aikaisemmassa vaiheessa projektia virheet löytyvät, sitä alhaisemmat ovat projektin kustannukset [Eastbrook et al., 1998]. Muodollista määrittelyä käytettäessä projektin kustannukset keskittyvät projektin alkupäähän. Kuvassa 7 esitetään, miten muodollisen määrittelyn käyttäminen vaikuttaa projektin kustannuksiin.



Kuva 7. Muodollisen määrittelyn vaikutukset kustannuksiin. [Sommerville, 2007]

Kuvasta 7 huomataan, että tavallista prosessia käytettäessä vahvistuskustannukset ovat noin 50% kustannuksista. Suunnittelu- ja toteutuskustannukset taas ovat noin kaksi kertaa suuremmat kuin määrittelykustannukset. Muodollista määrittelyä käytettäessä toteutus- ja suunnittelukustannukset laskevat hieman ja ovat yhtä suuret määrittelykustannusten kanssa. Kuvasta 7 huomaamme myös, että vahvistamiskustannukset ovat alentuneet merkittävästi. Tämä johtuu siitä, että muodollinen määrittely poistaa ristiriitoja ja virheitä tehokkaasti, joten korjaustyötä ei tarvitse tehdä niin paljon vahvistusvaiheessa.

6. Nopea ohjelmistokehitys ja ketterät menetelmät

Kuten viidennessä luvussa käviilmi, ohjelmistoprojektilta vaaditaan usein tiukkaa aikataulua ja mahdollisimman nopeaa järjestelmän toimittamista. Tämän päivän liiketoimintamalli vaatii yrityksiltä sopeutumista nopeasti muuttuvaan ympäristöön ja globaaliin kilpailuun. Tästä syystä asiakkaalle on usein tärkeää saada järjestelmä mahdollisimman nopeasti käyttöön, jotta pystytään vastaamaan nopeasti muuttuvien markkinoiden tuomiin haasteisiin ja uusiin mahdollisuuksiin. Asiakas on usein jopa valmis tinkimään järjestelmän laadusta, jotta järjestelmän toimitus onnistuisi nopeammin. Nopea järjestelmän kehittäminen ja toimittaminen onkin tänä päivänä yksi kriittisimmistä vaatimuksista, joita ohjelmistotuotantoprojektilta vaaditaan [Sommerville, 2007].

Nopea ohjelmistokehitys on prosessimalli, jossa järjestelmä saadaan mahdollisimman aikaisessa vaiheessa asiakkaan käyttöön [Pressman, 2007]. Tämä on iteratiivinen prosessi, missä projektin eri vaiheet toteutetaan osittain yhtä aikaa. Koko järjestelmää ei toteuteta kerralla ja toimiteta lopuksi asiakkaalle. Nopeassa ohjelmistokehityksessä järjestelmä toteutetaan osissa. Ensin pyritään toteuttamaan järjestelmän päätoiminnallisuudet ja saamaan ne asiakkaalle käyttöön mahdollisimman nopeasti. Tämän jälkeen järjestelmää kehitetään pidemmälle ja toiminnallisuuksia lisätään asiakkaan järjestelmään vaiheittain sitä mukaa, kun järjestelmän osiot valmistuvat [Sommerville, 2008]. Kun järjestelmän osia saadaan käyttöön, tarkentuvat asiakkaan vaatimukset muille osioille.

Perinteisessä projektimallissa muutokset vaatimuksiin lisäävät kustannuksia, varsinkin projektin myöhemmissä vaiheissa. Nopean ohjelmistokehityksen menetelmät taas toimivat hyvin muuttuvien vaatimusten kohdalla [Coram and Bohner, 2005]. Muuttuviin vaatimuksiin pystytään reagoimaan nopeasti ilman, että se aiheuttaisi suurempia viivästyksiä tai lisäkustannuksia. Nopean kehityksen menetelmissä jopa oletetaan, että vaatimukset tulevat tarkentumaan ja muuttumaan.

Nopea ohjelmistokehitys ei ole kuitenkaan pelkästään varautumista muutoksiin ja tehokasta muutosten toteuttamista. Nopean kehityksen menetelmien tärkeimpiä ominaispiirteitä on sujuva kommunikointi [Pressman, 2005]. Nopean ohjelmistokehityksen menetelmissä tiimit ovat yleensä hyvin yhtenäisiä ja kommunikointi on helppoa ja nopeaa. Asiakas on hyvin tiiviissä yhteistyössä kehittäjien kanssa ja saattaa olla jopa osa tiimiä. Näin ollen kommunikointi asiakkaan ja kehittäjien välillä on vaivatonta.

Yksi nopean ohjelmistokehityksen ongelma on projektin hallinnointi. Nopean kehityksen projekteissa muutoksia tulee paljon, joten kovin tarkka dokumentointi projektissa ei yleensä ole kustannustehokasta [Sommerville, 2007]. Sopimuksen tekeminen asiakkaan kanssa järjestelmän toteuttamisesta voi olla

myös ongelmallista. Koska prosessi ei ole ennalta tarkkaan suunniteltu, tarkentuvat vaatimukset, prosessin kesto sekä työmäärä vasta projektin myöhemmissä vaiheissa. Näin ollen voi olla hankalaa tehdä kiinteää sopimusta asiakkaan kanssa projektin alkuvaiheessa.

Järjestelmän kustannusten kannalta, yksi suurimmista ongelmista on järjestelmän ylläpidettävyys. Koska järjestelmä toteutetaan vaiheittain ja toiminnallisuuksia lisätään jälkikäteen, voi järjestelmän rakenteesta tulla hyvin monimutkainen ja vaikeasti ymmärrettävä. Tämä tekee järjestelmän ylläpitämisestä hyvin vaikeaa, koska ylläpitäjä ei välttämättä ole ollut mukana järjestelmän kehityksessä.

Näistä ongelmista johtuen on paljon projekteja, joihin nopea kehitys ei ole paras vaihtoehto. Esimerkiksi kriittisissä järjestelmissä nopea ohjelmistokehitys on huono vaihtoehto, koska projektin täytyy olla erittäin huolellisesti suunniteltu, jotta se tulee onnistumaan [Coram and Bohner, 2005]. Kaikki vaatimukset tulee löytää ja niiden täytyy olla tarkkoja sekä yksiselitteisiä.

Ketterät menetelmät ovat malleja, joiden avulla nopeaa ohjelmistokehitystä voidaan toteuttaa. Tunnetuimman ketterät menetelmät ovat Scrum- ja Extreme-ohjelmointi [Salo and Abrahamsson, 2008]. Scrum on iteratiivinen malli, jossa järjestelmä toteutetaan ja mahdollisesti toimitetaan asiakkaalle osissa. Scrumin idea on, että projektin jäsenet tapaavat usein ja käyvät läpi, mitä ollaan toteutettu ja mitä tullaan toteuttamaan. Iteraatioita kutsutaan sprinteiksi ja kunkin sprintin aikana toteutetaan sillä hetkellä tärkeimmät toiminnallisuudet. Scrum keskittyy aktiiviseen projektin hallintaan, missä lyhyet sprintit minimoivat riskejä ja kasvattavat projektin onnistumisen todennäköisyyttä [Salo and Abrahamsson, 2008].

Extreme-ohjelmoinnissa kaikki vaatimukset esitetään skenaarioina ja työtehtävät toteutetaan järjestyksessä [Sommerville, 2007]. Ohjelmoijat työskentelevät pareittain ja jokaista työtehtävää pyritään arvioimaan ja testaamaan jo ennen sen toteuttamista. Kuten muissakin ketterissä menetelmissä, myös Extreme-ohjelmoinnissa prosessimalli koostuu iteraatioista ja järjestelmä toteutetaan osissa. Kun Scrum keskittyy enenmmän projektin hallintaan, keskittyy Extreme-ohjelmointi enemmän itse järjestelmän toteutukseen liittyviin toimintoihin [Salo and Abrahamsson, 2008].

Vaikka nopea ohjelmistokehitys ei toimi suoraan kaikissa projekteissa, on sitä kuitenkin mahdollista hyödyntää osittain laajemmissa ja monimutkaisissa projekteissa. Yksi tapa tähän on niin sanotun hybridimallin käyttö, joka koostuu perinteisen ohjelmistokehitysprosessin ja nopean ohjelmistokehityksen piirteistä. Esimerkiksi prototyyppien käyttö on toimiva tapa tehostaa vaatimusten ymmärtämistä, kun nopeaa ohjelmistokehitystä halutaan soveltaa monimutkai-

sempaan projektiin [Sommerville, 2007]. Prototyyppien avulla sekä asiakas että kehittäjät pystyvät ymmärtämään paremmin, mitä täytyy tehdä ja mitkä ovat vaatimukset. Prototyypin avulla voidaan myös varmistua siitä, että järjestelmä vastaa sille asetettuja vaatimuksia.

Prototyyppien käytöllä voi olla kaksi eri tarkoitusta. Sitä voidaan käyttää niin sanottuun osittaiseen kehittämiseen, missä on tarkoitus aloittaa järjestelmän toteuttaminen niistä toiminnoista, joiden vaatimukset ymmärretään parhaiten [Sommerville, 2007]. Nämä ovat yleensä myös järjestelmän päätoiminnot. Epäselvät ja vähemmän tärkeät vaatimukset toteutetaan vasta, kun ne ymmärretään paremmin.

Niin sanotun Throw-away -prototyypin käytön tarkoituksena on tuottaa ja vahvistaa järjestelmävaatimuksia [Sommerville, 2007]. Tässä mallissa aloitetaan vaatimuksista, jotka ovat epäselviä. Tarkoituksena on ymmärtää ja tarkentaa epäselvät vaatimukset, eikä päätoiminnoista tarvitse välttämättä tehdä prototyyppiä, koska vaatimukset ovat selvät.

Vaikka prototyyppien käyttö hidastaa projektia hieman, parantaa se järjestelmän käytettävyyttä ja ennen kaikkea käyttäjävaatimuksia ilman, että projektin kustannukset kasvavat huomattavasti [Sommerville, 2007]. Prototyyppien käyttö lisää kustannuksia projektin alkuvaiheessa, mutta vähentää kustannuksia projektin loppuvaiheessa. Tämä johtuu siitä, että vaatimukset ymmärretään paremmin jo varhaisessa vaiheessa projektia.

Jotta itse prototyypistä voidaan tehdä kustannustehokas, on harkittava tarkkaan, mitä osioita järjestelmästä otetaan mukaan prototyyppiin. Kaiken toiminnallisuuden huomioonottaminen ei välttämättä ole tarpeellista, joten prototyypissä kannattaa toteuttaa vain tärkeät ja kriittiset toiminnallisuudet [Sommerville, 2007]. Tämä sekä nopeuttaa projektin aikataulua että alentaa projektin kustannuksia.

7. Yhteenveto

Kuten tutkielman alussa todettiin, on vaatimusten määrittely hyvin haastava ja aikaa vievä prosessi. Tämä prosessi on kuitenkin syytä suorittaa huolella, sillä vaatimusmäärittelyvirheiden aiheuttamat toimintahäiriöt sovelluksessa voivat tulla asiakkaalle hyvin kalliiksi.

Vaatimusten määrittelyyn käytettävät tekniikat ja mallit riippuvat hyvin paljon projektin luonteesta. Voidaan todeta, että mikäli kyseessä on kriittinen järjestelmä, on vaatimusten määrittely syytä toteuttaa äärimmäisellä tarkkuudella ja huolellisuudella. Näin voidaan taata järjestelmän toimintavarmuus ja estää toimintavirheiden tapahtuminen.

Hyvä keino kriittisten järjestelmien vaatimusten määrittelyyn ja vaatimusten vahvistamiseen ovat muodolliset menetelmät. Muodollisten menetelmien avulla virheet ja ristiriidat löytyvät suurella varmuudella ja näin saavutetaan ohjelmiston hyvä laatu. Muodolliset menetelmät ovat verrattain hidas ja kallis prosessi, mutta asiakas on usein valmis investoimaan tähän, koska järjestelmän toimintahäiriö voi aiheuttaa erittäin suuria rahallisia menetyksiä tai jopa katastrofin.

Koska projektin kustannukset ovat sidoksissa projektiin kuluvaan aikaan, on asiakkaalle usein tärkeää, että järjestelmä pystytään toimittamaan mahdollisimman nopealla aikataululla. Järjestelmä halutaan usein saada myös käyttöön mahdollisimman nopeasti globaalin kilpailun ja nopeasti muuttuvien markkinoiden takia.

Liiketoimintaa tukevien järjestelmien toteuttamisessa kustannustehokas menetelmä on käyttää nopeaa ohjelmistokehitystä ja notkeita menetelmiä. Näin asiakas saa jo aikaisessa vaiheessa projektia järjestelmän päätoiminnot käyttöönsä ja vaatimusten määrittely tarkentuu sitä mukaa, kun projekti etenee. Muutosten tekeminen järjestelmään on helppoa ja verrattain halpaa, joten vaatimusten ei tarvitse olla alusta alkaen täysin yksiselitteisiä, vaan ne voivat tarkentua tai jopa muuttua projektin myöhemmissä vaiheissa.

Viiteluettelo

[Sommerville, 2007] Ian Sommerville, *Software Engineering*, 8th ed. Addison-Wesley, 2007.

[Kotonya and Sommerville, 2004] Gerald Kotonya and Ian Sommerville, *Requirements Engineering: Processes and Techniques*. Wiley, 2004

[Pressman, 2005] Roger S. Pressman, *Software Engineering: A Practitioner's Approach,* 6th ed. McGraw-Hill, 2005.

[Attarha and Modiri, 2011] Mina Attarha and Nasser Modiri, Focusing on the importance and the role of requirement engineering. In: *Proc. of 4th Int. Conf. on Interaction Sciences (ICIS)*, 2011, 181–184.

[Easterbrook et al., 1998] Steve Easterbrook, Robyn Lutz, Richard Covington, John Kelly, Yoko Ampo and David Hamilton, Experiences using lightweight formal methods for requirements modeling. *IEEE Transactions on Software Engineering* **24**, 1 (1998), 4–14.

[Hamilton et al., 1995] David Hamilton, Rick Covington and Alice Lee, An experience report on requirements reliability engineering using formal methods. In: *Proc. of 6th Int. Symp. on Software Reliability Engineering, 1995,* 52–57.

[Salo and Abrahamsson, 2008] Outi Salo and Pekka Abrahamsson, Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum. *Software, IET* **2**, 1 (2008), 58–64.

[Coram and Bohner, 2005] Michael Coram and Shawn Bohner, The Impact of Agile Methods on Software Project Management, In: *Proc. of 12th IEEE Int. Conf. on Engineering of Computer-Based Systems (ECBS '05), 2005*, 363–370.

Prosessi avoimen lähdekoodin käyttöjärjestelmässä

Nikita Ishkov

Tiivistelmä.
Tämä tutkielma tutustuttaa lukijan tietokonekäyttöjärjestelmien sisäisen toiminnan periaatteeseen. Tarkemmin ottaen yhteen käyttöjärjestelmän osaan – prosessien vuorontajaan. Prosesseilla tarkoitetaan tietokoneella ajettavia ohjelmia ja vuoronnuksella näiden ohjelmien hallintaa.

Avainsanat ja -sanonnat: Linux, prosessi, säie, vuorontaja.
CR-luokat: D.4.1

1. Johdanto

Tutkielman aiheena on prosessien vuoronnus avoimen lähdekoodin käyttöjärjestelmissä. Työssä tarkastelun kohteena ovat yleiset periaatteet, joita nykyiset järjestelmät noudattavat. Alussa esittelen aihetta vain yleistetyllä abstraktiotasolla ja tutkielman viimeisessä luvussa näytän, miten nämä abstraktiot toimivat todellisuudessa. Työn toinen luku kertoo käyttöjärjestelmistä, niiden historiasta ja siitä, miten päädyttiin nykyiseen järjestelmien rakenteeseen. Tämän jälkeen määritellään, mikä on prosessin olio tietokonekäyttöjärjestelmässä, mistä se koostuu ja miten se käyttäytyy; tutustutaan säikeisiin. Seuraavaksi siirrytään työn pääaiheeseen, prosessien vuoronnukseen. Viimeksi tarkastellaan Linux-käyttöjärjestelmän ensimmäisen version vuorontajan toimintaa.

2. Käyttöjärjestelmistä yleensä

Käyttöjärjestelmistä puhuttaessa ajatellaan yleensä jonkinlaista ikkunointijärjestelmää, mustavihreää komentokehotetta tai vastaavaa. Tämä on luonnollista, sillä käyttöliittymä on usein ainoa käyttöjärjestelmän osa, jonka käyttäjä huomaa. Kuitenkin tämä mielikuva on väärä. Käyttöjärjestelmä sisältää selvästi enemmän kuin (pseudo-)grafiikan. Se on iso kokonaisuus, jonka päätehtävä on toimia liitäntänä ohjelmien ja laitteiston välillä, jolloin syntyy rajapinta, joka antaa mahdollisuuden käynnistää ja ajaa erilaisia sovelluksia, mukaan lukien käyttöliittymä.

Käyttöjärjestelmä on siis ohjelmisto, joka toimii sovellusohjelmien ja laitteiston välillä [Haikala ja Järvinen, 2004]. Tätä ohjelmistoa kutsutaan myös ytimeksi. Tästä määritelmästä seuraa se tosiasia, että ydin tarvitsee laitteiston tukea ja että se sisältää laitteistosta riippuvaista koodia pystyäkseen toimimaan. Tässä tapauksessa laitteistolla tarkoitetaan suoritinta ja sen arkkitehtuuria. Arkkitehtuuri siis määrittelee sen, mitä suoritin tarjoaa järjestelmän ohjelmoijalle, eli miten käytännössä ydin näkee tämän suorittimen: rekistereiden määrä, konekäskyt, osoitusmuoto jne. Se, miten ytimen sisällä toteutetaan jokin toiminnallisuus, jää ohjelmoijan tehtäväksi.

Raudan päälle rakentuu niin sanottu ”hierarkkinen kone” [Haikala ja Järvinen, 2004], sillä tietokonejärjestelmä voidaan ajatella koostuvan kerroksista, joista kukin pohjautuu edellisen kerroksen tarjoamaan rajapintaan. Tähän asti on määritetty kaksi kerrosta: laitteisto ja käyttöjärjestelmän ydin ja niiden välinen rajapinta, suoritinarkkitehtuuri. Seuraavaksi hierarkiassa tulee käyttöjärjestelmän ytimen tarjoama virtuaalikonerajapinta, jota erilaisten ohjelmointikielten kääntäjät — seuraava kerros — voivat käyttää. Kääntäjä puolestaan määrittelee ohjelmointikielen struktuurin (rajapinnan): käskyjen kannan ja syntaksin. Sovelluksia kirjoitetaan ohjelmointikielellä, jonka kääntäjä ”puhuu” käyttöjärjestelmälle virtuaalikoneen välityksellä. Käyttöjärjestelmä puolestaan ohjaa rautaa arkkitehtuurin käskyillä.

Miksi näin monimutkaista? Miksi ei sallita sovelluksen ohjata rautaa suoraan, ilman lukuisia abstraktioita? Vastaus tähän kysymykseen seuraa käyttöjärjestelmien historiasta, jota esitellään lyhyesti seuraavassa.

Ensimmäisien sukupolven tietokoneissa [Tanenbaum, 2008] ei ollut lainkaan sellaista käsitettä kuin käyttöjärjestelmä. Silloin ei tiedetty ohjelmointikielistäkään, ei edes symbolisesta konekielestä. Ohjelmointi tapahtui konekielellä (binäärimuoto) ja muodostamalla virtapiirejä. Tällaisen koneen suorittimen toiminnan algoritmi on yksinkertaistettuna seuraavanlainen:

1. hae muistista osoite, johon käskylaskuri viittaa
2. kasvata käskylaskuria yhdellä (yhden käskyn verran)
3. lue haetusta muistiosoitteesta käsky (jokin käsky suorittimen käskykannasta)
4. suorita käsky (kirjoita muistiin, lue muistista, hyppää osoitteeseen, lue reikäkortti...).

Ideana on se, että ohjelma koostuu käskyistä, jotka suoritetaan järjestyksessä niin kauan, kunnes ne loppuvat. Koneen muistipaikkojen sisältö asetetaan ulkoisilla kytkimillä. Kytkimillä syötetään myös binäärimuodossa oleva latauS-

ohjelma, joka lataa muistiin suoritettavan ohjelman. Itse ohjelma luetaan esimerkiksi reikäkorteilta.

Edellisen sukupolven tietokoneessa ei ollut massamuistilaitetta ollenkaan, vaan se koostui muistista ja suorittimesta (von Neumannin mukainen arkkitehtuuri). Lisätään nyt (1. sukupolvi, yksiajokäyttöjärjestelmä) [Haikala ja Järvinen, 2004] olemassa olevaan malliin oheislaitteet: kovalevy, kortinlukija ja kirjoitin. Pyritään automatisoimaan ohjelman rutiinit laatimalla yksinkertainen käyttöjärjestelmä. Oletetaan, että kortinlukija lukee reikäkortteja pakasta niin kauan, kunnes pakka on tyhjä, eli pakassa voi olla useita ohjelmia peräkkäin. Jokainen uusi ohjelma alkaa symbolilla, jonka käyttöjärjestelmä huomaa, ja myös loppuu symboliin niin, että järjestelmä osaa erottaa eri ohjelmat toisistaan. Koneen käynnistyksen jälkeen käyttöjärjestelmä lukee kortinlukijalta ohjauskomennon (esim. #START), tämän jälkeen käyttöjärjestelmä lukee ohjelman koodin, jonka se lataa kovalevylle ja syöttää kääntäjälle. Kääntäjä kääntää ohjelman. Seuraava ohjauskomento ajaa ohjelman (esim. #RUN) ja tarvittaessa ohjaa tulosteen kirjoittimeen. Ohjelman suorituksen lopettaa ohjauskomento (esim. #STOP), ja järjestelmä lataa seuraavan työn, joka alkaa komennolla (#START). On selvää, että käyttöjärjestelmä pitää sisällään vähintään koodin kovalevyllä olevien tiedostojen käsittelyä varten ja kääntäjän.

Minkälaisia ongelmia tässä huomataan? Ensinnäkin, jos käyttöjärjestelmän koodi sijaitsee muistissa, jokainen ohjelma voi kirjoittaa sen päälle tai lukea sieltä, esimerkiksi virheellisen ohjelmakoodin seurauksena. Näin käyttöjärjestelmä voi tuhoutua tai, vielä huonommin, ohjelma antaa väärän tulostuksen. Toinen ongelma on ikuiset silmukat, taas ohjelmakoodin virhe voi pysäyttää ajojonon ilman, että tulostusta odottava henkilö saa tietää asiasta.

Tässä huomataan, että käyttöjärjestelmän tärkeäksi tehtäväksi nousee resurssien hallinta. Muistiin liittyvä ongelma voidaan ratkaista asettamalla järjestelmän koodin raja muistiavaruudessa. Kuvataan koko muistiavaruus sanoilla (yhden sanan pituus vaikkapa 2 tavua). Silloin koko muisti on 0 — N-1 sanaa. Asetetaan ydin alueelle 0 — 10, kutsutaan sitä aluetta ytimen avaruudeksi. Nyt ohjelmat saavat käyttää vain välillä 11 — N-1 olevaa muistia. Mikä estää ohjelmaa kirjoittamasta ytimen avaruuteen tai muuttamasta sen rajaa? Lisätään järjestelmän koodiin muuttuja USER_MODE vastaamaan siitä, että käyttäjän tilassa ajettava ohjelma ei pystyisi kirjoittamaan tai lukemaan ytimen avaruutta. Muuttujan ollessa tilassa 1 järjestelmä on käyttäjän tilassa, ja kun USER_MODE-muuttujan arvo on 0, järjestelmä on ytimen tilassa [Rodriguez *et al.*, 2005]. Täytyy määritellä vielä käsky, joka saa muuttaa USER_MODE-muuttujan arvoa. Luonnollisesti sen voi suorittaa vain ydin, toisin sanoen tämä

on etuoikeutettu käsky. Käyttäjän tilassa oleva ohjelma aiheuttaa virheen (virhekeskeytyksen), jos yrittää muuttaa USER_MODE-muuttujan arvoa tai kirjoittaa tai lukea ytimen muistia (asettaa käskylaskurin osoitteeseen 0-10). Virhekeskeytys yleensä tarkoittaa ajettavan ohjelman välitöntä pysähtymistä ja logiikan siirtymistä ytimen avaruuteen, jossa käyttöjärjestelmä päättää, miten toimia. Käytännössä minkä tahansa etuoikeutetun käskyn kutsu käyttäjän tilasta aiheuttaa virhekeskeytyksen (SIGSEGV-signaali) [Mitchell, 2001].

Ikuinen silmukka on vaarallinen sen takia, että se syö koko suorittimen ajan, eli järjestelmä ei siirry seuraavaan käskyyn ennen kuin edellinen on suoritettu loppuun. Tämän ongelman ratkaisemiseksi lisätään laitteeseen kello, josta vähennetään ykkönen tasaisin aikavälein. Järjestelmään kovakoodataan jokin arvo — kellon maksimiaika. Suoritus palaa ytimen avaruuteen heti, kun kello nollautuu. Tätä kutsutaan kellokeskeytykseksi.

Edellisen sukupolven järjestelmä sallii vain yhden työn ajon kerrallaan. Ajossa oleva sovellus ei kuitenkaan kuluta kaikkia saatavilla olevia resursseja. Yli 90 prosenttia ajasta suoritin joutuu odottamaan luettavia reikäkortteja nopeallakin kortinlukijalla. Myös keskusmuisti ei aina ole kokonaan käytetty pienen ohjelman ajettaessa. Näin syntyy kysymys, miten otetaan vapaat resurssit käyttöön. Vastaus on johdonmukainen: suorittamalla useita ohjelmia samanaikaisesti. Tämä on moniajokäyttöjärjestelmän idea (2. sukupolvi, moniajokäyttöjärjestelmä) [Haikala ja Järvinen, 2004].

Tässä tulee vastaan hyvin samanlainen ongelma kuin yksiajojärjestelmässä: miten suojataan eri suorituksessa olevien ohjelmien eli prosessien muistialueet toisistaan? Ratkaisuna tähän voidaan käyttää samanlaista menetelmää kuin ytimen avaruuden tapauksessa. Yhden rajan sijasta käytetään vaan kahta. Ensimmäinen (BASE) viittaa prosessin muistialueen alkuun, toinen (LIMIT) sisältää alueen pituuden. Nyt jokaisen prosessin osoiteavaruus on nollasta alkava osoitteiden BASE-LIMIT välinen alue [Rodriguez *et al.*, 2005]. Koko muistin kannalta, eli todellisen muistin osoitteet prosessin osoiteavaruudessa on tämän prosessin muistiavaruus. Edellinen pitää paikkansa silloin, kun USER_MODE-muuttuja on tilassa 1, kun ollaan ytimen tilassa, BASE-rekisteri asetetaan arvoon 0 ja LIMIT arvoon N-1. Näin käyttöjärjestelmä saa viitata mihin tahansa muistiosoitteeseen ja käyttäjän prosessi vain omaan alueeseensa.

Yksiajojärjestelmässä määritetty kellokeskeytys tulee nyt hyvin tärkeäksi moniajojärjestelmän osaksi. Kaikki ajossa olevat ohjelmat eivät voi olla suorituksessa samaan aikaan yksisuoritinjärjestelmässä, ja niinpä yhden prosessin ajossa olon aika on rajoitettava. Oletetaan tässä, että kaikki prosessit ovat yhtä kiireellisiä, eli haluavat päästä suoritukseen yhtä nopeasti. Sanotaan, että niillä

on sama prioriteetti [Haikala ja Järvinen, 2004]. Nyt laitekellon on tasoitettava prosessien kuluttava suoritinaika. Tämä tapahtuu kellokeskeytyksen avulla: prosessi pääsee ajoon, kuluttaa aikansa (sanotaan 100ms), minkä jälkeen tapahtuu keskeytys ja suoritus tuttuun tapaan siirtyy ytimen avaruuteen, jossa käyttöjärjestelmä valitsee seuraavan prosessin ajettavaksi. Tätä toimintoa kutsutaan skeduloinniksi [Haikala ja Järvinen, 2004]. Seuraavan työn (prosessin) valitseva käyttöjärjestelmän osa on yleensä pieni aliohjelma, sitä kutsutaan vuorontajaksi (scheduler). Mitä nyt tapahtuu, kun vanha prosessi menettää oikeutensa käyttää suoritinta ja uusi tulee tilalle? Siirtyessä ytimen avaruuteen järjestelmä asettaa uudet rajat muistiin ja ylikirjoittaa suorittimen rekistereiden arvot. Näin kaikkien odotustilassa olevien prosessien tiedot (esim. LIMIT- ja BASE- muuttujien arvot) tallennetaan muistiin. Kun prosessi taas pääsee suoritukseen, nämä tiedot haetaan ja sovellus jatkaa toimintaansa normaalisti. Tällaista prosessikohtaisten tietojen tallennusta ja hakua kutsutaan ympäristön vaihdoksi (context switch).

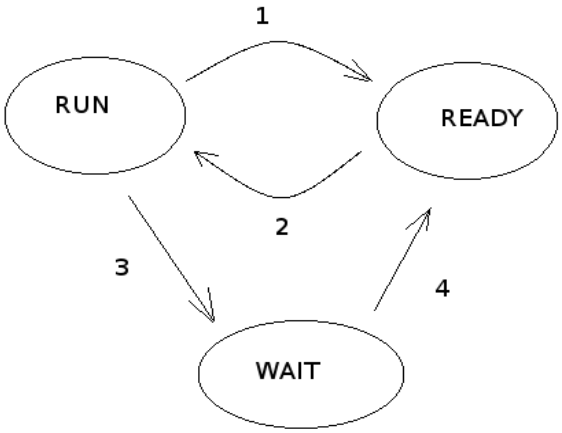
3. Prosessit ja säikeet

Tässä luvussa tarkastelen prosesseja yksityiskohtaisesti. Niin kuin todettiin aikaisemmin, prosessi on suorituksessa oleva ohjelma. Yksi käyttöjärjestelmän ytimen tärkeimmistä tehtävistä on aloittaa, ohjata ja lopettaa prosesseja. Edellä on karkeasti näytetty, miten yksi suoritin voi pitää ajossa useita sovelluksia niin, että järjestelmän toiminnan aikana näyttää kuin esim. internet-selain, sähköpostiohjelma ja musiikkisoitin toimivat yhtä aikaa eli rinnakkain. Nykyisten suorittimien nopeuden ja isojen muistimäärien ansiosta prosessien ympäristön vaihto tapahtuu niin nopeasti, että satoja sovelluksia voivat toimia käyttäjän mielestä ”rinnakkain”. Tätä kutsutaan näennäiseksi (pseudo/quasi parallelism) [Tanenbaum, 2008] rinnakkaisuudeksi, eli yhtenä tiettynä ajanjaksona on ajossa vain ja ainoastaan yksi prosessi. Aito rinnakkaisuus toimii vain silloin, kun järjestelmässä on enemmän kuin yksi suoritin. Tässä tutkielmassa tarkastellaan kuitenkin prosessien käyttäytymistä yksisuoritinjärjestelmässä ja siihen liittyvää prosessien vuoronnusta.

Tarkastelun kohteena voidaan pitää seuraavanlaista mallia: järjestelmän toiminta perustuu vuorontajaan, joka ohjaa joukkoa prosesseja. Nämä prosessit voivat olla joko käyttäjän luomia tai itse käyttöjärjestelmän rutiineja.

3.1. Prosessien tilamalli

Käyttöjärjestelmässä prosessi voi olla suorituksessa (RUN), odottamassa suoritinta (READY) tai odottamassa jotain tapahtumaa (WAIT) [Tanenbaum, 2008], esimerkiksi levylle kirjoittamista tai syöttöä käyttäjältä. RUN-tilassa oleva työ on siis parhaillaan käyttämässä suoritinta. Silloin, kun tapahtuu kellokeskeytys ja prosessi luovuttaa suorittimen toisen prosessin käyttöön vasten tahtoaan (irrottava vuoronnus) [Haikala ja Järvinen, 2004], siirtyy tämä prosessi tilaan READY. READY-tilassa ovat siis kaikki prosessit, jotka odottavat pääsyä suoritukseen. Vuorontajan päätehtävä on valita READY-tilasta RUN-tilaan menevän prosessin. WAIT-tilaan siirtyvät työt, jotka eivät voi jatkaa toimintansa ilman jotakin järjestelmän tai laitteiston resurssia. Yleisimmät WAIT-tilassa olevat prosessit ovat kirjoitus/luku -operaatiota odottavat työt tai interaktiiviset sovellukset, jotka vuorovaikuttavat käyttäjän kanssa. Interaktiiviset prosessit eivät yleensä käytä koko niiden saamaa aikaviipaletta (quantum) [Bach, 1986], vaan siirtyvät heti odottamaan syöttöä WAIT-tilaan, ei-interaktiiviset taas käyttävät suoritinta raskaasti. Prosessien tilamalli on esitetty kuvassa 1.



Kuva 1. Prosessien tilamalli.

Luonnollisesti prosessi siirtyy tilasta toiseen elämänsä aikana. Kuten kuvassa 1 on esitetty, siirtoja on neljä erilaista. Siirrot 1 ja 2 ovat vuorontajan ohjaamia ja ne tapahtuvat ikään kuin väkivaltaisesti, ilmoittamatta prosessille mitään. Siirto 1 tapahtuu silloin, kun prosessi on kuluttanut koko aikaviipaleensa, kello nollautuu ja vuorontaja ohjaa työn READY-tilaan odottamaan seuraavaa suoritukseen pääsyä. Siirto 2 ohjaa vuorontajan valitseman työn suoritukseen silloin, kun kyseinen työ on odottanut tarpeeksi pitkään. Se, miten vuorontaja valitsee seuraavaksi suoritukseen pääsevän työn, on tämän tutkielman keskeinen aihe ja

sitä tarkastellaan myöhemmin. Silloin, kun käyttöjärjestelmä päättää, että jokin työ ei tällä hetkellä tarvitse suoritinta, tapahtuu siirto 3. Siinä tapauksessa prosessi yleensä odottaa jotakin. Kun tämä prosessi saa odottamansa resurssin, tapahtuu siirto 4 ja työ pääsee odottamaan ajoon pääsyä. Jos sattuu niin, että mikään työ ei ole RUN-tilassa, on hyvin todennäköistä, että WAIT-tilasta tullut työ pääsee suoritukseen siirron 2 välityksellä heti. [Bovet and Cesati, 2005]

Edellä kuvattu prosessien tilamalli auttaa ymmärtämään, mitä tapahtuu käyttöjärjestelmän ”konepellin” alla, vaikka todellinen tilanne on paljon monimutkaisempi, esimerkiksi kaikkia prosessien tiloja ei edes mainita. Tässä tutkielmassa esitetään vain ne perusasiat, jotka helpottavat ymmärtämään, miten käyttöjärjestelmän vuorontaja toimii. Linux-käyttöjärjestelmässä tilat RUN ja READY on yhdistetty tilaan TASK_RUNNING [Bovet and Cesati, 2005]. Tämä tarkoittaa, että prosessi ei ole WAIT-tilassa.

3.2. Prosessien toteutus

Kun käynnissä saattaa olla satoja sovelluksia yhtä aikaa, miten käyttöjärjestelmä ylläpitää kaikkia ajossa olevia prosesseja? Ydin pitää sisällään jokaista prosessia kohti niin sanotun prosessielementin (task_struct) [Rodriguez *et al.*, 2005], johon kerätään prosessikohtaisia tietoja kuten BASE- ja LIMIT- muuttujien arvot, käskylaskuri, prosessin avattujen kovalevyllä olevien tiedostojen deskriptorit jne. Toisin sanoen kaiken tärkeän informaation, joka täytyy tallentaa silloin, kun prosessi siirtyy suorituksesta odotus- tai valmiustilaan, jotta se voisi jatkaa suoritusta ikään kuin sitä ei olisi koskaan pysäytetty. Yksilöivä elementti prosessikentässä on yleensä järjestysnumero. Prosessielementit muodostavat «prosessitaulun» (task_list). Itse asiassa se ei ole taulukko, Linux-ytimessä se on syklinen kahteen suuntaan linkitetty lista (doubly linked list).

Nyt meillä on tarpeeksi tietoja sitä varten, että voimme määritellä, miten prosessien vaihto tapahtuu käyttöjärjestelmässä. Näin tulee samalla määriteltyä, miten muodostuu näennäinen rinnakkaisuus. Muistiavaruudessa sijaitsee osoite, johon järjestelmän suoritus ”hyppää” keskeytyksen sattuessa (ainoa määritelty keskeytys meillä on kellokeskeytys), kutsutaan sitä keskeytysvektoriksi (interrupt vector) [Bovet and Cesati, 2005]. Yksisuoritinjärjestelmän ytimen avaruuden tapaan jokin määrä muistin alkuosoitteista on varattu keskeytysten prosessointia varten. Olkoon prosessi A käyttänyt koko saamansa aikaviipaleen suorittimessa ja tapahtui kellokeskeytys. Laitteisto (kello kuuluu laitteistoon, eikä ohjelmistoon) tallentaa ensin kyseisen prosessin käskylaskurin ja tilan, ja ohjaus siirtyy käyttöjärjestelmälle. Järjestelmä tallentaa joitain elämäntärkeitä prosessielementin kenttiä. Tämän jälkeen logiikka siirtyy keskeytysvektoriin,

etsii sieltä tapahtuneen keskeytyksen koodin ja luovuttaa ohjauksen keskeytyskäsittelijälle. Meidän tapauksessamme kellokeskeytys kutsuu vuorontajan, joka alkaa toimintansa ja päättää, mikä prosessi pääsee suoritukseen seuraavana. Uuden prosessin valittua vuorontaja välittää logiikan käyttöjärjestelmälle, joka hakee kyseisen prosessin numeron prosessitaulusta ja tuo tämän prosessin kaikki tiedot esiin rekistereihin, ja uusi prosessi aloittaa toimintansa.

3.3. Säikeet

Edellä kuvatussa käyttöjärjestelmän mallissa yhdellä prosessilla voi olla yksi ja ainoa järjestys, jossa suoritin lukee ja toteuttaa ohjelman käskyjä sekä yksi osoiteavaruus tätä prosessia varten. Aikaisemman määritelmän mukaan yksi ajettava ohjelma muodostaa yhden prosessin. Joskus kuitenkin tarvitaan useita erillisiä työjonoja yhden prosessin sisällä, joita suoritetaan (näennäisesti) rinnakkain ja jotka käyttävät samaa muistia osoiteavaruudessa. Esimerkkinä voi olla mikä tahansa ohjelma, jonka sisällä on suorituksessa useita tehtäviä ja osa niistä on aina välillä estettynä. Näitä useita tehtäviä yhden prosessin sisällä kutsutaan säikeiksi (threads, light-weight processes) [Haikala ja Järvinen, 2004]. Joissakin järjestelmissä (Linux) vuorontaja toimii vain säikeiden kanssa. Säie onkin pienin skeduloiva yksikkö käyttöjärjestelmässä, ja jokainen prosessi koostuu ainakin yhdestä säikeestä [Haikala ja Järvinen, 2004].

Säikeiden käyttöön on monta syytä. Tarkastelemme muutamia tässä. Yksi tärkeimmistä säikeiden tuottamista eduista on se, että sellaiset kevyet prosessit käyttävät yhtä ja samaa muistia (osoiteavaruutta) ja samoja avoinna olevia tiedostoja yhden ison prosessin sisällä. Säikeen vaihtuessa prosessin sisällä ei tarvitse vaihtaa BASE- ja LIMIT- muuttujien arvoja, tallentaa tiedostojen deskriptoreita jne. Tarvitsee vain siirtyä uuteen muistilohkoon, joka sijaitsee samassa osoiteavaruudessa. Käytännössä tämä säästää huomattavasti aikaa verrattuna siihen, että vaihdetaan kokonaisia raskaita prosesseja. Toinen hyvä syy liittyy säikeiden nopeaan luontiin ja tuhoamiseen verrattuna isoihin prosesseihin. Useimmissa järjestelmissä säikeen luonti tapahtuu 10-100 kertaa nopeampi kuin uuden kokonaisen prosessin käynnistäminen [Tanenbaum, 2008]. Tämä ominaisuus on tarpeen silloin, kun säikeiden määrä kasvatetaan ja vähennetään dynaamisesti nopealla tahdilla. Lopulta säikeet sallivat rinnakkaislaskennan yhden sovelluksen sisällä monisuoritinjärjestelmässä, jossa on olemassa todellinen mahdollisuus aitoon rinnakkaisuuteen.

Prosesseja käytetään yhdistämään tietokoneen resursseja ajettavia ohjelmia varten ja säikeet ovat vuorontajan suoritukseen ohjaamia olioita. Linux-käyttöjärjestelmässä tämä on vain abstraktio, todellisuudessa vuorontaja ei näe

eroa prosessien ja säikeiden välillä. Ydin näkee säikeitä (tekstissä esiintyvät myös nimellä «tehtävä», task) erillisinä prosesseina ja vuorontaa niitä prosessien tapaan (kellokeskeytyksen tapahtuessa). Kuten aikaisemmin todettiin, prosessi koostuu vähintään yhdestä tehtävästä (task), eli alussa (oletetaan, että sovellus lisää tehtäviä suorituksen aikana ja käynnistyy yhdellä säikeellä) sovelluksessa on yksi säie. Sitä sanotaan ryhmän johtajaksi (group leader) [Bovet and Cesati, 2005], ja se saa tämän prosessin yksilöivän numeron (itse asiassa se on säikeiden ryhmän yksilöivä numero (thread group id, tgid)) ja säikeen yksilöivän numeron (thread id, tid). Silloin, kun prosessiin lisätään tehtäviä, uusi säie liittyy ryhmään ja saa ryhmän johtajan id:n yksilöivän tunnuksen kenttään task_struct-prosessielementissa ja oman yksilöivän säikeen numeron (tid). Ohjelmoija voi hakea ryhmän id:n getpid() kutsulla (get process id). Näin edellä kuvaamani prosessiabstraktio Linux-järjestelmässä on ikään kuin ryhmä säikeitä (thread group) ja säie ikään kuin kevyt prosessi. UNIX-järjestelmissä prosessit muodostavat hierarkian: parent — children (siblings).

Klassinen UNIX-käyttöjärjestelmän prosessin luontitapa on sellainen, että suorituksessa oleva prosessi luo itsestään kopion, eli kopioi kaiken käyttämänsä muistin ja kaikki käytössä olevat resurssit. Sitten tämä uusi kopio (lapsiprosessi) yleensä puhdistaa saamansa kopion muistista ja aloittaa oman tehtävänsä normaalisti. On selvää, että tällainen käytäntö on ajan tuhlausta. Niinpä modernit UNIX-ytimet, kuten Linux, tarjoavat nopeampia ratkaisuja [Bovet and Cesati, 2005];

- Copy on write -tekniikka. Lapsiprosessi saa oman kopion parent-prosessin muistista vain silloin, kun se haluaa ylikirjoittaa sitä, eli turhaa kopiointia ei tehdä. Käytännössä on harvinaista, että lapsi haluaisi muokata parent-prosessin muistia.
- Säikeet käyttävät johtajasäikeen resursseja.
- vfork()-systeemikutsu luo uuden kopion prosessista, joka käyttää samoja resursseja. Kopion ollessa suorituksessa parent-prosessi on lukittu siihen asti, kun lapsi lopettaa tai käynnistää uuden sovelluksen.

Prosessi päättyy (kuolee) yleensä itse silloin, kun se lopettaa tehtävänsä. Kuoleman yhteydessä prosessi ilmoittaa tapahtumasta ytimelle, niin, ja järjestelmä voi vapauttaa työn pitämiä resursseja. Ydin voi myös tuhota koko ryhmän säikeitä, jos yksikin säie tässä ryhmässä saa käskyn käyttäjältä (interaktiivisessa sovelluksessa) tai vaikkapa yrittää suorittaa kielletyn komennon. Kannattaa huomata, että lapsiprosessit kuolevat ennen niiden parent-prosesseja, koska parent-prosessi sulkee ja vapauttaa kaikki lapsen käyttämät resurssit. Jos parent kuolee ennen sen lapsia, lapset muuttuvat init-prosessin lapsiksi, joka

tuhoaa ne. Init-prosessi on UNIX-käyttöjärjestelmissä kaikkien prosessien äiti, se on ensimmäisenä ajossa järjestelmän käynnistettyä ja se synnyttää kaikki muut prosessit, sen tunnus (pid) on yleensä 1. [Bovet and Cesati, 2005]

4. Skedulointi

Kaikki prosessit käyttöjärjestelmässä voidaan karkeasti jakaa kahteen pääosaan: suorittimen nopeuteen rajoittuneet ja syöte- ja tulostuslaitteiden nopeuteen rajoittuneet (myös interaktiiviset prosessit, jotka vuorovaikuttavat käyttäjän kanssa). Suorittimen nopeuteen rajoittuneet prosessit käyttävät keskussuoritinta koko niiden saaman aikaviipaleen verran ja ovat suurilta osin ei-vuorovaikutteisia, esimerkkinä video- tai audiomuuntaja tai muu raskaita laskuja suorittava sovellus. Syöte- ja tulostuslaitteiden nopeuteen rajoittuneella ohjelmalla tarkoitetaan paljon esimerkiksi massamuistilaitetta käyttävää sovellusta. Sellainen prosessi viettää huomattavasti enemmän aikaa odottamassa esimerkiksi levyltä lukemista tai levylle kirjoittamista eikä käytä suoritinta yhtä aktiivisesti kuin joku raskaita laskuja tekevä soveluus. [Rodriguez *et al.*, 2005]

Tarkastellaan tilannetta, jossa käyttäjä pelaa shakkipeliä. Ohjelman logiikka on jaettu kahteen osaan: yksi osa odottaa käyttäjän vuoroa, toinen laskee seuraavia siirtoja. Mitä tapahtuisi, jos syöttöä odottava osa käyttäisi aina koko saamansa suorittimen aikaviipaleen? Jokaisen käyttäjän vuoron jälkeen ohjelma joutuisi miettimään seuraavaa siirtoa. Tämä olisi käytännössä hidasta (hitaalla laitteistolla) ja käyttäjän tyytyväisyys peliin olisi alhainen. Niinpä prosessien vuoronnuksella käyttöjärjestelmässä on hyvin tärkeä rooli. Käyttöjärjestelmän on siis huolehdittava siitä, että suoritin (sekä kaikki muut laitteiston osat) on mahdollisimman tehokkaasti käytössä. Tämä saadaan aikaan vaihtamalla suorituksessa olevia prosesseja optimaalista algoritmia käyttäen. Jokainen vuorontaja siis käyttää jonkinlaista vuoronnusalgoritmia. Optimoinnin lisäksi vuoronnusalgoritmin tulisi olla reilu (fair); mikään prosessi ei saisi olla suorituksessa liian pitkään ja mikään prosessi ei saisi odottaa vuoroansa liian pitkään.

Skedulointi voi olla kahdenlaista: prosessien vuoronnus ja säikeiden vuoronnus [Tanenbaum, 2008]. Edellinen tarkoittaa tilannetta, jossa yhden prosessin (sovelluksen) sisällä on useita työjonoja. Silloin ytimen vuorontaja vuorontaa vain prosessiolioita ja sovellus hoitaa säikeiden vaihdot itse. Tällainen lähestymistapa sallii sovelluksen ohjata omia tehtäviä niin, että se ei käytä keskeytyksiä, vaan yksi säie voi käyttää niin paljon prosessille annettua aikaa kuin se haluaa. Näin ohjelmoija saa päättää miten säikeet vuorovaikuttavat keskenään. Tässä työssä tarkastellaan toista tapaa, eli säikeiden skedulointia ytimen tilassa,

missä säikeet vaihtuvat kellokeskeytyksen tapahtuessa (pre-emptive). Sellainen on käytössä Linux-ytimessä.

5. Vuoronnusmenetelmiä

Tässä luvussa tarkastellaan yksinkertaisia vuoronnusmenetelmiä. Todellisuudessa mitään seuraavista ei käytetä puhtaana, vaan erilaisia menetelmiä yhdistetään ja saadaan tilanteeseen sopiva (paras) ratkaisu.

5.1. Ei-prioriteettiset menetelmät

FIFO (first in first out) [Tanenbaum, 2008]. Säikeet pääsevät suoritukseen siinä järjestyksessä kuin ne ovat jonossa READY-tilassa odottamassa. Tässä ei käytetä kellokeskeytyksiä, vaan suorituksessa oleva työ luopuu suorittimesta vain silloin, kun se siirtyy WAIT-tilaan odottamaan jotain tapahtumaa. Tällöin seuraava pääsee suoritukseen. Ensimmäisen työn palattua READY-tilaan, se ohjataan jonon loppuun. Tämä on suhteellisen reilu menetelmä sekä helppo toteuttaa. Huonoja puolia ovat seuraavat: Ei ole sellaista käsitettä kun prioriteetti, eli kiireellinen säie joutuu odottamaan yhtä kauan kuin kaikki muutkin. Interaktiivisissa sovelluksissa ongelmana on hidas vasteaika. Ikuinen silmukka pysäyttää kaikki muutkin säikeet.

Round robin -vuoronnusmenetelmä [Haikala ja Järvinen, 2004] on samanlainen kuin FIFO, mutta siinä käytetään kellokeskeytyksiä. Näin jokainen säie saa olla suorituksessa korkeintaan sille annetun aikaviipaleen verran. Sen loputtua säie siirtyy READY-tilaan jonon loppuun. Algoritmin hyvä puoli on sen reiluus: kaikki työt odottavat saman verran. Ongelman muodostaa aikaviipaleen pituus. Nimittäin, jos se on liian lyhyt, esim. 10 ms, ja säikeen vaihto kestää 2 ms, silloin noin 20 % suorittimen ajasta menee hukkaan. Se on liian paljon. Toisaalta, jos aikaviipale on liian pitkä, sanotaan 200 ms, ja suoritukseen pääsee yhtäkkiä 50 kiireellistä työtä, toinen järjestyksessä oleva työ ei ala ennen kuin 202 ms on kulunut. Viimeinen saa odottaa liiankin pitkään.

Lyhin työ ensin -menetelmässä [Tanenbaum, 2008] oletetaan, että kaikkien prosessien suoritusajat tiedetään ennalta. Jos päästetään suoritukseen ensin kaikki pienemmät työt, saadaan tehokkaampi algoritmi. Tarkastellaan esimerkkiä, jossa meillä on 3 prosessia, joiden suoritusajat ovat 2, 2 ja 10 minuuttia. Jos suoritukseen pääsee 10 min prosessi, sen jälkeen kaksi muuta, saadaan suoritusajat: ensimmäiselle työlle 10 min, toiselle 12 ja kolmannelle 14. Keskiarvo 12 min. Jos käytetään ”lyhin työ ensin” algoritmia, niin ensimmäisen työn suoritusaika on 2 min, toisen 4 min ja viimeisen 14 min. Keskiarvo 6,6. Kolmen pro-

sessin suoritukseen käytetty keskiaika melkein puolittuu. Ongelmana tässä menetelmässä on tietenkin se, miten kerätään tietoja prosessien suoritukseen tarvittavista ajoista. Varma ratkaisu tähän on arvioida aikaisempia suorituksia.

5.2. Prioriteettiset menetelmät

Kiinteän prioriteetin menetelmässä [Haikala ja Järvinen, 2004] kaikille säikeille annetaan kiinteä prioriteetti. Suoritukseen pääsee se, joka on prioriteetiltaan korkeampi. Menetelmä on hyvin yksinkertainen toteuttaa ja se määrittää säikeiden välillä tärkeysjärjestyksen. Haittapuolena on menetelmän epäreiluus. Tietyissä tilanteissa alempiprioriteettiset säikeet voivat nälkiintyä (starve). Korkeampiprioriteettinen säie voi jopa keskeyttää keskeytyskäsittelijän.

Vaihtelevan prioriteetin menetelmässä [Tanenbaum, 2008] korkeampiprioriteettisten säikeiden ongelman ratkaisemiseksi vuorontajan pitää alentaa niiden prioriteettia jollakin tavalla, esimerkiksi jokaisen kellokeskeytyksen yhteydessä. Kun ajossa olevan prosessin prioriteetti on pienempi kuin jonkun muun, tapahtuu prosessien vaihto. Toinen keino on vaihtaa prosesseja kellokeskeytyksen yhteydessä niin, että ajoon pääsee aina se työ, jolla on korkein prioriteetti jonossa olevista. Tällöim READY-tilassa täytyy olla vähintään kaksi jonoa: RUN-tilasta saapuneet (out) ja RUN-tilaan menevät (in). Kun in-jono tyhjenee, kaikki prosessit siirtyvät out-jonosta takaisin in-jonoon ja ajoon pääsee taas se työ, jolla on korkein prioriteetti).

Usein on tarpeen ryhmitellä prosesseja prioriteetin perusteella. Näin kaikki saman prioriteetin työt sidotaan samaan ryhmään ja ryhmien välinen vuoronnus tapahtuu jollakin prioriteettisella menetelmällä. Ryhmän sisällä olevat työt noudattavat round robin -menetelmää. Vuoronnus näyttää seuraavanlaiselta: ensin suoritukseen pääsee ryhmä, jolla on korkein prioriteetti ryhmien välillä, esimerkiksi 10. Kaikki ryhmässä olevat prosessit saavat aikaviipaleensa vuoronperään. Silloin, kun ryhmän kaikki työt lopettavat, ajoon pääsee ryhmä, jonka prioriteetti on seuraava, esimerkiksi 9.

Viimeisenä menetelmänä tarkastellaan reilua vuoronnusta [Tanenbaum, 2008]. Tähän asti tarkastelun kohteena oli vain joukko prosesseja ja niiden vuoronnus. Todellisuudessa ajossa voi olla yksi prosessi, jolla on paljon lapsiprosesseja ja ne kaikki ovat yhden käyttäjän töitä ja vielä yksi toisen käyttäjän työ. Näin ensimmäinen käyttäjä saa paljon enemmän suoritinaikaa kuin toinen. Tämä ei ole reilua. Jotkut käyttöjärjestelmät ottavat tämän huomioon ja lupaavat jokaiselle käyttäjälle saman verran tietokoneen resurssien käyttöaikaa.

6. Linux

Linux-käyttöjärjestelmän ensimmäisen version (0.01) vuorontaja oli yksinkertainen, hidas ja muutenkin rakenteeltaan erilainen verrattuna monimutkaisempiin nykyajan vuorontajiin. Skedulerin koodi oli myös helppo toteuttaa ja ymmärtää. Tässä luvussa selitän sen toimintaa. Lisäksi tarkastelen joitain hyödyllisiä funktioita sched.c-tiedostosta. Tässä luvussa lukijalta vaaditaan C-ohjelmointikielen osaamista sekä tietokoneen arkkitehtuurin perusteiden tuntemusta.

6.1. Linux 0.01 ja sen vuorontaja

Ytimen vuorontaja löytyy kansiosta 'kernel', päätiedosto on nimeltään sched.c. Aluksi määritellään joitakin muuttujia.

Kellokeskeytysten välinen tauko: `#define LATCH (1193180/HZ)`. Luku 1193180 (laitteiston oskillaattorin taajuus = 1.19318 MHz) jaettuna sadalla (meidän tapauksessa HZ on asetettu arvoon 100), `#define HZ 100,` sched.h; tämä on laitteiston kellon keskeytysten määrä sekunnissa. Nykyisellä laitteistolla se on 1000 tai 1024).

Koodikatkelmassa 1 määrittelyn unionin avulla on mahdollista käsitellä prosessielementtiä pinosegmenttinä (stack segment) tai itse prosessielementtinä.

```
union task_union {
      struct task_struct task;
      char stack[PAGE_SIZE];
};
```

Koodikatkelma 1. Unioni task_struct.

task_struct unionin ensimmäinen olio on init_task (Koodikatkelma 2), joka on kaikkien task_struct-prosessielementtien kahteen suuntaan linkitetyn listan alku (ja loppu, lista on syklinen) ja samalla idle-prosessin prosessielementti. Idle-prosessi on niin sanottu nollaprosessi. Käyttöjärjestelmä suorittaa sitä silloin, kun mikään muu prosessi ei tarvitse suoritinaikaa.

```
static union task_union init_task = {INIT_TASK,};
```

Koodikatkelma 2. Nollaprosessi.

Globaali muuttuja jiffies `long volatile jiffies=0;` pitää sisällään kellon keskeytysten määrän järjestelmän käynnistyksestä lähtien. Muuttujaa kasvate-

taan joka kerta, kun tapahtuu kellokeskeytys. Avainsanalla volatile estetään kääntäjän optimoinnit [Banahan, 1991].

Pointteri `current struct task_struct *current = &(init_task.task)` viittaa tällä hetkellä suorituksessa olevaan prosessiin. Se luodaan ja alustetaan tässä idle-prosessin osoitteella. Pointteri, joka viittaa prosessiin, joka on viimeksi käyttänyt FPU:ta luodaan ja alustetaan null pointterilla `*last_task_used_math = NULL;`.

Prosessielementteihin viittaamien pointtereiden taulukko luodaan ja alustetaan idle-prosessin osoitteella. (`#define NR_TASKS 64, sched.h). struct task_struct * task[NR_TASKS] = {&(init_task.task), };`.

Funktio void math_state_restore() (Koodikatkelma 3) tarkistaa, onko viimeisin prosessi käyttänyt FPU:ta. Jos on käyttänyt, se tallentaa FPU:n tilan. Tallentamiseen on käytetty symbolisen konekielen funktiota fnsave() [Hyde, 1996]. Jos tällä hetkellä suoritettava prosessi on käyttänyt FPU:ta ennen keskeytystä, funktio lataa talletetun FPU:n tilan (assemblyn frstor()). Jos se ei ole käyttänyt, FPU:n tila alustetaan (fninit()) ja tämänhetkisen prosessin ”onko käyttänyt FPU:ta” -prosessielementin attribuuttiin asetetaan arvo 1. Pointteri ”viimeksi FPU:ta käyttänyt prosessi” alustetaan myös tämänhetkisen prosessin osoitteella.

```
void math_state_restore() {
      if (last_task_used_math)
            __asm__("fnsave %0"::"m" (
                  last_task_used_math->tss.i387));
      if (current->used_math)
            __asm__("frstor %0"::"m" (current->tss.i387));
      else {
            __asm__("fninit"::);
            current->used_math=1;
      }
      last_task_used_math=current;
}
```

Koodikatkelma 3. math_state_restore().

Vuorontajan pääfunktio on nimeltään schedule (void schedule(void)). Silloin kun työ lopettaa (tai siirtyy WAIT-tilaan) ja prosessit vaihtuvat, se kutsuu tätä funktiota. Tämä on itse asiassa rinnakkaisuutta toteuttava mekanismi. Funktio alkaa kommentilla «*’schedule()’ is the scheduler function. This is GOOD*

CODE! There probably won't be any reason to change this...». Pointteri ensimmäisen työn prosessielementtiin elementtien taulukossa on makro FIRST_TASK, viimeiseen — LAST_TASK (`#define FIRST_TASK task[0], #define LAST_TASK task[NR_TASKS-1],` sched.h).

Koodikatkelmassa 4 on esitetty itse vuorontajan koodi. Käydään se läpi rivi riviltä. Vuoronnuksen prosessi alkaa käymällä kaikki työt läpi (rivillä 4 alkava silmukka). Jokaisen prosessin kohdalla tarkistetaan, että pointteri ei viita nollaan (rivi 5) ja herätetään se lähettämällä SIGALRM-signaali, nollataan samalla työn herätyskello (rivit 6 – 9). Jos prosessi sai signaalin ja se voi vastaanottaa keskeytyksiä, siirretään prosessi TASK_RUNNING-tilaan (rivit 10 – 12). Riviltä 13 alkaa itse vuoronnus, kuten rivillä oleva kommentti kertoo. Käynnistetään ikuinen silmukka, joka lopetetaan silloin, kun löydetään prosessi, joka ei vielä käyttänyt aikaviipaleensa loppuun ja jonka aikaviipale on kaikista suurin. Jos ei löydetä prosessia, jonka aikaviipale on nollasta eroava, switch_to() funktio käynnistää nollaprosessin (rivit 13 – 32). Riveillä 19 – 26 käydään taas kaikki prosessit läpi ja etsitään suurimman aikaviipaleen työ. Jos kaikki prosessit ovat käyttäneet oman aikaviipaleensa loppuun, käydään vielä kerran kaikki (nukkuvat myös) prosessit läpi ja vaihdetaan niiden laskureiden arvot. Niille, joiden aikaviipale == 0 asetetaan counter = priority (prioriteetin arvo), muille: counter/2 + priority (jäljellä jäävä aikaviipale jaettuna kahdella plus prioriteetin arvo) (rivit 27 – 30). Täten prosessilla on herätessään entistä enemmän suoritinaikaa. Rivillä 32 käynnistetään joko suurimman aikaviipaleen työ tai nollaprosessi. Saattaa olla, että suoritukseen pääsee sama työ, joka kutsui 'schedule() funktiota'.

```
1 void schedule(void) {
2     int i,next,c;
3     struct task_struct ** p;
4     for(p = &LAST_TASK ; p > &FIRST_TASK ; --p)
5           if (*p)
6                 if ((*p)->alarm && (*p)->alarm < jiffies) {
7                             (*p)->signal |= (1<<(SIGALRM-1));
8                             (*p)->alarm = 0;
9                       }
10                if ((*p)->signal && (*p)->state==TASK_INTERRUPTIBLE)
11                      (*p)->state=TASK_RUNNING;
12          }
```

```
13    /* this is the scheduler proper: */
14    while (1) {
15          c = -1;
16          next = 0;
17          i = NR_TASKS;
18          p = &task[NR_TASKS];
19          while (--i) {
20                if (!*--p)
21                      continue;
22                if ((*p)->state ==
23                      TASK_RUNNING && (*p)->counter > c)
24                            c = (*p)->counter, next = i;
25          }
26          if (c) break;
27          for(p = &LAST_TASK ; p > &FIRST_TASK ; --p)
28                if (*p)
29                      (*p)->counter =
30                            ((*p)->counter >> 1) + (*p)->priority;
31    }
32    switch_to(next);
33 }
```
Koodikatkelma 4. Vuorontaja.

Seuraavaksi tarkastellaan esimerkin vuoksi yhtä etuoikeutettua käskyä (Koodikatkelma 5). Ytimen tilassa oleva prosessi (sys_komento tarkoittaa etuoikeutettua käskyä) saa itse mennä nukkumaan asettamalla itselleen tilan, josta se reagoi signaaleihin ja kutsulla schedule(). Kun tämä prosessi saa signaalin ja haluaa vastata siihen, schedule() herättää sen; return 0 lopettaa sys_pause()'n toiminnan ja prosessi jatkaa normaalisti.

```
int sys_pause(void) {
      current->state = TASK_INTERRUPTIBLE;
      schedule();
      return 0;
}
```
Koodikatkelma 5. sys_pause().

Kellokeskeytysten käsittelijä kutsuu do_timer()-funktiota (Koodikatkelma 6). Tämä funktio vaihtaa (kutsuu schedule()) käyttäjien prosesseja kellokeskeytyksen tapahtuessa. Jos cpl (current privilege level) on asetettuna (cpl == 1), kasvatetaan utime-nimistä prosessielementin kenttää (user time) (rivit 3 – 4), eli prosessi on käyttäjän tilassa. Jos cpl == 0, prosessi on ytimen (etuoikeutetussa) tilassa. Kasvatetaan stime-kenttää (system time) (rivit 5 – 6). Funktion päätehtävä (rivit 7 – 10) on vähentää prosessin aikaviipaletta. Jos vähennyksen jälkeen se on vieläkin suurempi kuin nolla, funktio lopettaa. Jos ei ole suurempi kuin nolla ja jos tämä on käyttäjän prosessi, niin seuraa kutsu schedule(). Jos kyseessä on käyttöjärjestelmän prosessi, funktio lopettaa ja kontrolli palautetaan ytimen tilassa olevalle työlle.

```
1 void do_timer(long cpl)
2 {
3     if (cpl)
4           current->utime++;
5     else
6           current->stime++;
7     if ((--current->counter)>0) return;
8     current->counter=0;
9     if (!cpl) return;
10    schedule();
11 }
```

Koodikatkelma 6. do_timer().

Tässä ei näytetä, miten ydin suorittaa itse prosessien vaihdon (context switch). FPU:n tilan tallentaminen ja palauttaminen muistuttaa tätä ja se on samalla tavalla vain joukko 386-konekäskyjä, joka ei meitä kiinnosta.

Kuten nähdään, vuorontajan aikavaatimus riippuu prosessien määrästä järjestelmästä, joten asymptoottisessa notaatiossa [Cormen *et al.*, 2009] tämän vaatimus olisi O(n), eli lineaarinen aika.

Ensimmäisen Linux-ytimen vuorontajan koodi on helppo ymmärtää ja toteuttaa. Tämä ei kuitenkaan tarkoita, että se on paras mahdollinen. Vuorontajalla on ollut paljon ongelmia, esimerkiksi jos järjestelmässä on maksimimäärä prosesseja ja hidas suoritin, vuorontaja voi helposti syödä kaiken suorittimen ajan prosessien vaihdossa [Jones, 2006].

7. Yhteenveto

Tässä työssä tarkastellut menetelmät kuvaavat alustavasti käyttöjärjestelmien toimintaa koskien prosessien hallintaa. Käytännössä tilanne saattaa olla hieman erilainen, mutta lähestymistapa on aina sama ja nämä perusasiat pysyvät jokaisen toteutuksen runkona. Kannattaa pitää kuitenkin mielessä, että työn kohteena ovat olleet vain avoimen lähdekoodin käyttöjärjestelmät, kuten Linux ja Minix.

Viiteluettelo

[Bach, 1986] Maurice J. Bach, *The Design of the UNIX Operating System, 1st edition*. Prentice-Hall, 1986.

[Banahan *et al.*, 1991] Mike Banahan, Declan Brady, and Mark Doran, *The C Book*. Addison-Wesley 1991.

[Bovet and Cesati, 2005] Daniel P. Bovet and Marco Cesati, *Understanding the Linux Kernel, 3d ed.* O'Reilly Media, 2005.

[Cormen *et al.*, 2009] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms. Third Edition.* The MIT Press, 2009.

[Haikala ja Järvinen, 2004] Ilkka Haikala ja Hannu-Matti Järvinen, *Käyttöjärjestelmät.* Talentum, 2004.

[Hyde, 1996] Randall Hyde, *The Art of Assembly Language*. Retrieved from http://oopweb.com/Assembly/Documents/ArtOfAssembly/Volume/toc.html. Checked November 19, 2011.

[Jones, 2006] Tim Jones, Inside the Linux scheduler. Developer works. Retrieved from http://www.ibm.com/developerworks/linux/library/l-scheduler/ Checked November 19, 2011.

[Mitchell *et al.*, 2001] Mark Mitchell, Jeffrey Oldham, and Alex Samuel, *Advanced Linux Programming*. New Riders Publishing, 2001.

[Rodriguez *et al.*, 2005] Claudia Salzberg Rodriguez, Gordon Fischer, and Steven Smolski, *The Linux Kernel Primer: A Top-Down Approach for x86 and PowerPC Architectures.* Prentice-Hall, 2005.

[Tanenbaum, 2008] Andrew S. Tanenbaum, *Modern Operating Systems. Third Edition.* Prentice Hall, 2008.

Tietämyksen muodostaminen käyttäjien käyttäytymisestä sähköisessä kaupankäynnissä

Olavi Karppi

Tiivistelmä.
Tutkielman tarkoituksena on tehdä katsaus tietämyksen muodostamiseen sähköisessä kaupankäynnissä. Sähköisessä kaupassa on mahdollista seurata käyttäjien toimintaa ja käyttäytymistä. Tämän seurannan avulla voidaan päätellä ostotottumuksia, tulevia trendejä sekä muuta tietämystä. Muodostetun tietämyksen avulla on mahdollista tehostaa kaupankäyntiä. Tutkielmassa tutkitaan lyhyesti menetelmiä tältä osa-alueelta keskittyen kuluttajille suunnattuun Internetissä tapahtuvaan kaupankäyntiin.

Avainsanat ja -sanonnat: Sähköinen kaupankäynti, tiedonlouhinta, web usage mining.
CR-luokat: H.2.8, K.4.4

1. Johdanto

Tilastokeskuksen tilasto vuoden 2009 osalta kertoo Suomessa tapahtuneen sähköisen kaupan arvoksi 15 miljardia euroa. Tästä 19 prosenttia koostuu kotitalouksien tekemistä tilauksista. Tilastokeskuksen tutkimukseen osallistuneet yritykset saivat kaikkiaan 6,1 prosenttia liikevaihdostaan sähköisestä kaupasta. [Tilastokeskus, 2010] Vaikka kuluttajille suunnattu Internetin kautta tapahtuva sähköinen kauppa ei olekaan merkittävin tulonlähde useimmille yrityksille, on se kuitenkin sähköisen kaupan muodoista se, johon tavallinen kuluttaja useimmiten törmää.

Kuluttajille sähköinen kauppa tarjoaa vaivattoman tavan tehdä ostoksia, ja samalla se lisää mahdollisuuksia tuotteiden ja hintojen laajempaan vertailuun. Myyjäosapuolelle sähköinen kauppa tarjoaa kustannustehokkaan tavan tuotteiden myyntiin suuremmalle joukolle kuin mihin perinteisellä kivijalkamyymälällä on yleensä mahdollisuuksia.

Sähköinen kauppa mahdollistaa myös tehokkaan tavan kaupassa vierailevien asiakkaiden käyttäytymisen ja ostotottumusten selvittämiseen – jokainen sähköisessä kaupassa tapahtuva liike on mahdollista tallentaa analysointia varten.

Myyjäosapuolelle käyttäjien liikkeiden ja tottumusten selvittäminen tarjoaa mahdollisuuden tehostaa liiketoimintaa sekä palvella asiakkaitaan paremmin. Perinteisissä kivijalkaliikkeissä tottumuksia on voitu selvittää esimerkiksi videovalvonnan avulla. Videoiden läpikäyminen vaatii kuitenkin paljon ihmis-

työtä ja on siten helposti hidasta, kallista sekä virhealtista. [Kohavi, 2001] Sähköisessä kaupassa käyttäjien seurannasta syntyvä data on mahdollista tallentaa muotoon, joka sallii koneellisen prosessoinnin alusta alkaen ja helpottaa näin datan käsittelyä sekä analysointia.

Seurannasta saadun datan tiedonlouhinnalla ja analysoinnilla (web usage mining) voidaan selvittää esimerkiksi mahdollisia nousevia trendejä, puutteita tuotevalikoimassa tai muutostarpeita sivurakenteeseen. Analysoinnin pohjalta voidaan myös muokata markkinointistrategioita tai tarjota käyttäjille yksilöidympää sisältöä ja täten pyrkiä parantamaan käyttäjäkokemusta sekä lisäämään myyntiä. [Liu, 2011]

2. Tiedonlouhinta sähköisessä kaupassa

Tiedonlouhintaprosessi muodostuu sähköisessä kaupassa samanlaisista askeleista kuin muussakin tiedonlouhinnassa. Tarkastellaan sähköisen kaupan tiedonlouhintaa käyttäen pohjana vuonna 1999 julkaistua tiedonlouhintaprosessin kuvausta, CRISP-DM:ää (CRoss Industry Standard Process for Data Mining), joka kuvaa tiedonlouhintaprosessin yleisellä tasolla. CRISP-DM jaottelee prosessin kuuteen pääkohtaan [Chapman *et al.*, 1999]:

- liiketoiminnan sekä tavoitteiden ymmärtämiseen
- datan ymmärtämiseen
- datan esiprosessointiin
- tiedonlouhintaan / mallien muodostamiseen
- tulosten evaluointiin
- tiedonlouhinnasta saadun tietämyksen hyödyntämiseen.

Käydään nämä kuusi kohtaa tarkemmin läpi seuraavissa luvuissa.

2.1. Yleisesti tiedonlouhinnan tavoitteista

Sähköisessä kaupassa voi olla monia erilaisia tavoitteita tietämyksen muodostamisen kannalta. Samaan tapaan kuin perinteisissä kivijalkakaupoissakin, voi sähköisessä kaupassa kaupan muoto vaihdella kokonsa, tuotevalikoimansa, palveluvalikoimansa, kohderyhmänsä, maantieteelliseen toimialueensa sekä monen muun piirteen suhteen. Yhden miehen pienellä erikoistuotekaupalla on varmasti erilaiset tavoitteet kuin esimerkiksi Amazon.comilla, joka työllistää yli 30000 henkeä, toimii maantieteellisesti laajalla alueella ja jonka tuotevalikoima on hyvin laaja [YahooFinance, 2011].

Tietämyksen muodostaminen asiakkaiden käyttäytymisestä on väline, jolla asiakkaiden tarpeita sekä sivuston puutteita voidaan hahmottaa. Sivuston rakenne saattaa esimerkiksi olla niin hankala, ettei kävijä löydä haluamaansa tuotetta, tai voi olla, että jo pelkkä etusivu karkottaa suurimman osan kävijöistä.

Sähköisessä kaupassa on mahdollista tallentaa jokainen siirtymä sivulta toisella sekä aika, joka kullakin sivulla vietettiin. Sähköisessä kaupassa on helppoa seurata sitä, mistä ”ovesta” asiakas tuli kauppaan sisään, mitä reittiä asiakas ”ostoskärryään” kuljetti sekä mistä ”ovesta” poistuminen tapahtui. Jokainen siirtymä on mahdollista tallentaa, joten myös se saadaan selville, kauanko yhden ”hyllyn” tai tuotteen parissa aikaa vietettiin.

Tiedonlouhintaprosessin taustalla tavoitteina voi olla esimerkiksi kannattavuuden parantaminen, brändiarvon nostaminen tai asiakasuskollisuuden kasvattaminen. Näihin tavoitteisiin voidaan koettaa päästä saamalla sivustolle suurempia kävijämääriä tai saamalla sivustolle päätyvät kävijät viihtymään sivustolla paremmin. Tärkeää on myös, että käyttäjät löytävät sivustolta kaipaamansa asiat sekä palaavat sivustolle myöhemmin uudelleen.

Kannattavuutta voidaan parantaa kasvattamalla sitä osuutta kävijöistä, jotka tuotteita ostavat. Myös kävijämäärien kasvattamisella voidaan kasvattaa myyntimääriä, mikäli ostavien asiakkaiden määrä suhteessa kaikkiin kävijöihin pysyy vakiona. Myyntiä voidaan kasvattaa myös kasvattamalla kerralla myytyjä määriä tai myymällä tuotteita, joissa on paremmat katteet.

2.2. Personointi ja mainonta

Verkkokauppaan siirryttäessä voi käyttäjä avata kaupan joko kirjoittamalla osoitteen suoraan selaimen osoiteriville, käyttäen kirjanmerkkiä tai käyttäjä voi päätyä kauppapaikkaan joltain toiselta sivustolta. Mikäli käyttäjä päätyy sivustolle hakukoneen kautta, tallentuu lokeihin usein myös hakusanat, joilla sivustolle päädyttiin. Käytetyt hakusanat kertovat sen, mitä käyttäjä on ollut etsimässä ennen sivustolle päätymistä. Linkit muilta sivustoilta voivat kertoa kauppapaikan suosiosta esimerkiksi jossain tietyssä yhteisössä tai sosiaalisessa palvelussa. Linkkejä kauppapaikkaan voi olla esimerkiksi johonkin tiettyyn aiheeseen keskittyneellä keskustelualueella. Näiden tietojen pohjalta voidaan hahmottaa vierailijoiden piirteitä sekä miettiä sopivia paikkoja sivuston mainostamiselle.

Tiedonlouhinnan tavoitteena voi olla myös se, että sivustolla voitaisiin esittää käyttäjäkohtaisesti personoitua sisältöä. Hyvä esimerkki käyttäjäkohtaisesta personoinnista on Amazon.com, joka nostaa etusivulleen tuotteita, jotka liittyvät käyttäjän viimeksi ostamiin tai etsimiin tuotteisiin. Kauppapaikka saattaa siis näkyä eri käyttäjille erilaisena riippuen siitä, mitä sivustolla on viimeksi tehnyt. Myös erilaiset suosittelijajärjestelmät ovat tuttu näky useilla kauppapaikoilla. Suosittelijajärjestelmät ja sivuston personointi mahdollistavat sen, että käyttäjän on helpompi löytää uusia mielenkiintoisia tuotteita. Jotkin kaupat mukauttavat myös hintojaan sen mukaan, miten käyttäjä sivustoa on käyttänyt [McCarthy, 2000].

Tiedonlouhinnalla voidaan myös seurata mainonnan ja mainoskampanjoiden onnistumista. Käyttäjille voidaan lähettää toisistaan eroavia mainoksia ja tiedonlouhinnan avulla voidaan seurata näiden eri mainostyylien tehoa [Ansari *et al., 2001*].

Tietämyksen muodostamisella on siis mahdollista parantaa sähköistä kauppapaikkaa useilla eri tavoilla. Edellä esitetyt esimerkit kuvaavat muutamia näistä eri mahdollisuuksista. Ennen varsinaista tiedonlouhintaa on kuitenkin tärkeää ymmärtää, minkälaisesta liiketoiminnasta on kyse ja asettaa tavoitteet, jotka tiedonlouhinnan tahdotaan täyttävän. Näin tiedonlouhinnalle saadaan päämäärä, jota kohti pyrkiä ja johon tuloksia voidaan myöhemmissä vaiheissa verrata.

3. Datan kerääminen ja ymmärtäminen

Tiedonlouhinta on sähköisen kaupan tapauksessa lähes välttämätön työkalu, mikäli dataa tahdotaan ymmärtää paremmin. Tietomäärät ovat käytännössä aina niin suuria, ettei niiden ymmärtäminen ole mahdollista käymällä dataa läpi ”käsin”. Sähköisen kaupan tapauksessa sekä rivi- että attribuuttimäärät ovat usein suuria. Esimerkkinä attribuuttimääristä voidaan ottaa KDD Cupin julkaisema esimerkkidata [KDD Cup, 2000], joka kerättiin verkkokauppa Gazelle.comista. Tässä esimerkkidatassa on yli 500 attribuuttia. [Kohavi *et al.*, 2000] Lisäksi joidenkin attribuuttien erityispiirteet, kuten tuotekategorioiden hierarkkisuus, voi muodostaa omat haasteensa tiedonlouhintaan [Ansari *et al.*, 2001; Kohavi *et al., 2004*].

3.1. Datan keräämisestä

Sähköisessä kaupassa käyttäjien seuraamista voidaan toteuttaa useammalla erilaisella tavalla. Yksi vaihtoehto on seurata verkon läpi HTTP-palvelimelle kulkevaa liikennettä (packet sniffing) tai käyttää hyväksi HTTP-palvelimen lokitusta [Hussain *et al.*, 2010]. Molemmat tavat ovat hyviä siltä kannalta, että niitä on mahdollista käyttää lähes kaikissa sähköissä kaupoissa. Verkkoliikenteen seuranta ei tosin ole aina mahdollista. Seuranta estyy, mikäli esimerkiksi sivuston liikenne on salattua. Vaihtoehtoisia tapoja seurannan toteutukseen on käyttäjän suorittamien tapahtumien lokittaminen sovellustasolla (application level logging) [Ansari *et al.*, 2001] tai jonkin erillisen tunnistetiedoston (esimerkiksi kuvan) tai skriptin lisääminen jokaiselle sivulle ja tämän tunnisteen latausten seuraaminen [Kohavi, 2001].

Verkkoliikenteestä tai HTTP-palvelimen lokeista kerätty data kertoo käyttäjän liikkeistä hyvin yksinkertaisessa muodossa. Yksittäinen sivu koostuu useista resursseista, kuten kuvista sekä html-, tyyli- ja skriptitiedostoista. Tästä

seuraa, että yksittäinen sivun lataus aiheuttaa joukon tapahtumia, jotka kuvaavat näiden resurssien lataamista HTTP-palvelimelta käyttäjän selaimeen. Yksittäisten resurssien seuraaminen ei kuitenkaan yleensä ole mielenkiintoista, vaan tästä datasta täytyy pystyä erottamaan sivunlataukset.

Seurantatavasta riippumatta datasta pitäisi kuitenkin löytyä pääpiirteiltään samankaltaisia attribuutteja. Jokaisesta sivun latauksesta (linkin tai painikkeen painalluksesta) syntyy jälki, josta näkyy, mille sivulle siirryttiin ja mihin aikaan siirtymä tapahtui. Siirtymistä syntyy vierailuketju (click stream) [Liu, 2011], jota seuraamalla saadaan selville ne sivut, joilla käyttäjä on vieraillut, vierailujärjestys sekä yksittäisellä sivulla että koko sivustolla käytetty aika. Lisäksi samasta ketjusta saadaan selville aloitus- sekä lopetussivut; mikä oli käyttäjän ensimmäinen ja viimeinen sivu vierailun aikana.

Vierailuketjun perusyksiköksi voidaan mieltää yksittäinen sivun katselu (page view). Peräkkäiset yhden käyttäjän suorittamat sivun katselut muodostavat istunnon (session, activity record). Muutamia peräkkäisiä sivun katseluja kattava istunnon osajoukko on episodi (episode, business event). [Ansari *et al.*, 2001; Liu, 2011] Yhdeksi istunnoksi voidaan mieltää esimerkiksi etusivulle tuleminen, muutamien tuotteiden tietojen katselu, tuotteiden siirtäminen ostoskoriin sekä tilauksen tekeminen ja sivustolta poistuminen. Episodi on istunnon osajoukko, joka kattaa yksittäisen sivustolla suoritetun toimenpiteen. Episodi voi muodostua esimerkiksi yksittäisen tuotteen tietojen katselusta ja kyseisen tuotteen siirtämisestä ostoskoriin. Käyttäjästä saadaan muodostettua profiili, kun käyttäjän eri aikoina suorittamat istunnot kerätään yhteen.

3.2. Muu data

Sähköinen kauppa itsessään sisältää myös tietoa esimerkiksi sivuston rakenteesta, tuotteista, tuoteryhmistä sekä tuotteiden hinnoista. Sivujen katselut on tarpeen saada yhdistettyä näihin tietoihin. Pelkkä tieto siitä, että käyttäjä on käynyt tietyllä sivulla, ei välttämättä ole mielenkiintoinen. Mutta jos tähän yhdistetään tieto siitä, mitä ja minkälaisia tuotteita käyttäjä katseli, saadaan aikaiseksi mielenkiintoisempaa dataa. Mikäli datan kerääminen tehdään sovellustasolla, on tietojen yhdistäminen helpompaa [Ansari *et al.*, 2001].

Sivuston kävijöistä voidaan lisäksi kerätä dataa myös muilla tavoilla, kuten esimerkiksi kyselylomakkeiden avulla. Lokeista passiivisesti kerätty data ei välttämättä vastaa kysymyksiin siitä, miksi käyttäjä alun perin sivulle päätyi, miten käyttäjä koki sivuston ja miksi lopulta poistui sivustolta. Lokien pohjalta ei ole mahdollista selvittää käyttäjän ajatuksia. Lisäksi lokit eivät pysty vastaamaan siihen, pitikö asiakas ostamastaan tuotteesta tai miten toimitusketju toimi verkkokaupan ulkopuolella.

Jotkin sivustot tarjoavat mahdollisuuden vastata erilaisiin kyselylomakkeisiin sivustolla vierailun aikana. Tätä kautta on mahdollista saada tarkemmin selville esimerkiksi se, miksi käyttäjä tuli juuri kyseiselle sivustolle, mikä sivustolla oli hyvää tai löysikö käyttäjä etsimänsä tuotteen. Jotkin sivustot (kuten Amazon.com) lähettävät käyttäjille palautepyyntöjä ostotapahtuman jälkeen. Palautetta voidaan kysyä esimerkiksi muutama viikko ostotapahtuman jälkeen, jolloin on mahdollista kerätä tietoa käyttäjän tyytyväisyydestä ostetun tuotteen, toimituksen keston ja toimitustavan suhteen. Kun nämä palautteet liitetään osaksi muuhun kerättyyn dataan, voidaan käyttäjistä muodostaa tarkempi käyttäjäprofiili.

Myös muuta, sähköisen kaupan ulkopuolista, dataa voidaan hyödyntää. Dataa voidaan hankkia sähköisen kaupan ulkopuolisista lähteistä liittyen esimerkiksi väestön tai maantieteellisen alueen piirteisiin. [Liu, 2011]

4. Datan esiprosessointi

Datan esiprosessoinnilla tarkoitetaan kerätyn datajoukon siistimistä ja muokkaamista ennen varsinaista tiedonlouhintavaihetta. Esiprosessointi valmistelee datan tiedonlouhintamenetelmiä varten. HTTP-palvelinten lokien tiedonlouhinnassa yleisiä esiprosessointivaiheita ovat Hussainin ja muiden [2010] mukaan muun muassa datan siivoaminen (data cleaning), datan filtteröinti (data filtering), polkujen korjaaminen (path completion), käyttäjän tunnistaminen (user identification), istuntojen tunnistaminen (session identification) sekä istuntojen klusterointi (web session clustering). Mikäli data on kerätty sovellustasolla, ei näille kaikille vaiheille ole välttämättä tarvetta [Ansari *et al.*, 2001].

4.1. Datan siistiminen

Datasta voidaan siivota pois osia, joka eivät ole tiedonlouhinnan kannalta mielenkiintoisia. Mikäli tiedonlouhintaa tehdään HTTP-palvelimen lokien pohjalta, sisältävät lokit paljon ylimääräistä dataa, jota syntyy esimerkiksi kuva-, tyyli- ja skriptitiedostojen lataamisesta. Tämä on dataa, jolla ei useimmiten ole merkitystä tiedonlouhinnan kannalta ja joka on syytä siivota pois.

Muuta pois siivottavaa dataa voi sähköisen kaupan kohdalla olla esimerkiksi Internetissä toimivien hakurobottien (web spider) sivustolle tekemät automaattiset haut [Kohavi *et al.*, 2004]. Lisäksi sivustolle saattaa syntyä turhia sivunlatauksia myös tavallisten käyttäjien toimesta. Esimerkiksi Opera-selain tallentaa selaimen sulkemisen yhteydessä käyttäjän auki jättämät välilehdet ja lataa nämä välilehdet aina uudelleen, kun käyttäjä avaa selaimen. Tällaisissa tilanteissa sivustolle muodostuu sivunlatauksia, vaikkei käyttäjä välttämättä sivustolle itse menisikään. Sivunlataus muodostuu tällaisessa tilanteessa myös,

vaikkei käyttäjä edes katsoisi kyseistä välilehteä. Tällaisten sivunlatausten siivoaminen datasta voi olla hankalaa [Kohavi, 2001].

Datasta voidaan myös siivota ylimääräisiä attribuutteja pois. Yleisimmin HTTP-lokeista otetaan talteen vain käyttäjän IP-osoite, päivämäärä, aika, URL sekä selaimen tiedot (user agent) [Hussain *et al.*, 2010].

4.2. Muut esiprosessointivaiheet

Mahdollisia muita haasteita tiedonlouhintaan voivat aiheuttaa esimerkiksi WWW-selaimen sekä Internet-palveluntarjoajan välimuistit, jotka saattavat aiheuttaa sen, että sivulla käyminen ei aina jätä jälkiä palvelimelle. Vastaavasti selaimen back-painikkeen painallus ei kaikissa selaimissa nouda sivuja (joille ollaan palaamassa) palvelimelta, vaan esittää ne suoraan omasta välimuististaan. Usein tällaiset ongelmat saadaan kuitenkin kierrettyä täydentämällä vierailuketjua puuttuvat linkit täydentävän heuristiikan avulla [Liu, 2011].

Toisistaan erillisten istuntojen tunnistaminen vaatii rajanvedon siihen, kauanko kahden sivunlatauksen välillä voi olla viivettä ennen kuin sivun lataukset tulkitaan eri istuntoihin kuuluviksi. Yleisesti käytetty aika, jonka perusteella kaksi toisistaan erillistä istuntoa erotellaan, on 30 minuuttia [Hussain *et al.*, 2010].

Istuntojen tunnistamisen lisäksi datasta pitäisi pystyä tunnistamaan saman käyttäjän eri istunnot. Näin voidaan muodostaa käyttäjäprofiileja ja seurata sivustolle palaavien käyttäjien toimintaa. Tällä mahdollistetaan myös esimerkiksi sivuston käyttäjäkohtainen personointi.

Käyttäjän tunnistamista voidaan tehdä eri tavoin. Yhtenä vaihtoehtona on pakottaa käyttäjä kirjautumaan sivustolle sisään, mutta useimmiten kuluttajille suunnatuissa verkkokaupoissa tätä vaaditaan aikaisintaan ostoksia maksettaessa. Muita tapoja ovat muun muassa evästeiden tallentaminen käyttäjän koneelle sekä erilaiset tunnistustekniikat, joissa pyritään hyödyntämään selaimen lähettämiä yksilöiviä tietoja.

Electronic Frontier Foundation (EFF) on tutkinut sitä, miten yksilöivästi sivustolla vieraileva käyttäjä voidaan tunnistaa pelkästään selaimen lähettämien tietojen pohjalta. EFF:n edelleen jatkuvan tutkimuksen perusteella selaimeen pohjautuva käyttäjän tunnistaminen vaikuttaa jossain määrin mahdolliselta. Selaimen tiedot voidaan liittää käyttäjän IP-osoitteeseen, jolloin käyttäjän tunnistamisen todennäköisyyttä voidaan kasvattaa. [Eckersley, 2010]

Esiprosessointivaiheessa voidaan myös lisätä dataan uusia attribuutteja. Joitain attribuutteja voidaan myös muokata muihin formaatteihin. Tarkat päivämäärät ja kellonajat eivät esimerkiksi ole aina mielenkiintoisia, vaan dataa saatetaan haluta tarkastella päivä-, kuukausi- tai vuosi-tasolla, jolloin päivä-

määräkentistä on järkevää johtaa uudet näitä vastaavat attribuutit [Ansari *et al.*, 2001; Kohavi *et al.*, 2004].

Lisäksi voidaan johtaa myös täysin uutta dataa. Lokeista ei esimerkiksi saa selville sitä, jos osa asiakkaista ei vieraillut sivustolla viimeisen kuukauden aikana. Tiedonlouhinnan kannalta tämä voi kuitenkin olla mielenkiintoinen tieto, joka on järkevää lisätä datajoukkoon esiprosessointivaiheessa. [Chapman *et al.*, 1999]

Kerätty data on myös järkevää integroida muuhun olemassa olevaan dataan. Sähköisen kaupan tapauksessa tämä muu data koostuu esimerkiksi sivuston rakenteesta ja myynnissä olevien tuotteiden tuotetiedoista.

5. Tiedonlouhinta

Tiedonlouhinnan tarkoituksena on muodostaa ymmärrystä aiemmissa vaiheissa kerätystä ja esikäsitellystä datasta. Tiedonlouhinnan tavoitteena on muodostaa laajasta datajoukosta tietämystä, joka mahdollistaa datajoukon (tai sen osan) paremman ymmärtämisen. Yksinkertaisimmillaan sähköisen kaupan kohdalla datajoukosta voidaan päätellä esimerkiksi sivunlatausten kokonaismäärä tietyllä ajanjaksolla. Hieman laajennettuna esimerkkinä voidaan miettiä taulukkoa sivunlatausten jakautumisesta vaikkapa eri vuorokauden ajoille. Tulosten tulkintaa voidaan usein helpottaa tulosten visualisoinnilla, esimerkiksi kaksiulotteisesta taulukosta voitaisiin piirtää pylväsdiagrammi.

Kuvauksia, jotka kuvaavat koko datajoukkoa tai sen käyttäytymistä kutsutaan malleiksi. Datan pienempiä osajoukkoja kuvaavia kuvauksia kutsutaan hahmoiksi. Mallien ja hahmojen avulla voidaan perustella jo olemassa olevia oletuksia datasta, löytää uusia piirteitä tai pyrkiä ennustamaan tulevaa käyttäytymistä.

Tiedonlouhinnan lopputuloksena tulisi syntyä raportteja, visualisointeja sekä malleja [Ansari *et al.*, 2001; Kohavi *et al.*, 2004]. Kohavi ja muut [2004] suosittelevat, että tiedonlouhinta kannattaa aloittaa pienin askelin. Ensin pitää oppia kävelemään ennen kuin voi juosta, eikä tulosten tarvitse ensivaiheessa olla maailmaa mullistavia. Kohavi ja muut [2004] toteavat myös, että monimutkaisempia malleja ja tuloksia on usein hankalampi selittää ja perustella liiketoiminnan omistajalle, jolloin niiden hyöty jää usein pienemmäksi.

Raporttien avulla voidaan muodostaa yleiskuva kaupan ja sivuston toiminnasta. Mielenkiintoisia raportteja voivat olla esimerkiksi raportit myydyimmistä tuotteista, käytetyimmistä hakusanoista tai tuotteista, jotka siirretään ostoskoriin, mutta joita ei kuitenkaan lopulta osteta. Myös tiedot vierailluimmista sivuista, vierailuaikojen pituudet ja muut sivuston käyttöön liittyvät tiedot ovat yleisiä raporteissa [Liu, 2011]. Raporteista voidaan myös muodostaa sisältöä

sivustolle: listaus myydyimmistä tuotteista on yleinen näky sähköisissä kaupoissa.

Yleisiä tiedonlouhintamenetelmiä ovat muun muassa assosiaatiosääntöjen muodostaminen, datan luokittelu ja datan klusterointi [Liu, 2011]. Näiden menetelmien avulla voidaan johtaa monimutkaisempaa tietämystä kauppapaikan toiminnasta. Jos raporttien avulla voidaan vastata kysymykseen siitä, mitkä olivat myydyimmät tuotteet, voidaan tiedonlouhintamenetelmillä vastata esimerkiksi siihen kysymykseen, minkälaiset käyttäjät kyseisiä tuotteita ostavat ja minkälaiset piirteet johtavat ostotapahtumaan [Ansari *et al.*, 2001].

Assosiaatiosääntöjen avulla voidaan päätellä muun muassa, minkälaisia tuotteita ostetaan usein yhdessä toistensa kanssa. Samoin voidaan päätellä, minkälaiset käyttäjät ostavat tietynlaisia tuotteita. Klusterointimenetelmien avulla on mahdollista löytää samalla tavalla käyttäytyvää dataa. Klusterointi voi olla yksi työkalua esimerkiksi käyttäjäprofiilien löytämiseen ja mallintamiseen. Luokittelumenetelmät puolestaan tarjoavat välineitä, joilla voidaan päätellä, mihin ennalta määrättyyn kategoriaan data kuuluu. [Liu, 2011]

Kohavi [2000] toteaa, että on oleellista käyttää myös ihmismieltä tiedonlouhinnassa. Esimerkiksi mainoskampanjat tai tuotteiden poistuminen tuotevalikoimasta voivat vaikuttaa tuloksiin. Tiedonlouhintamenetelmät eivät kuitenkaan välttämättä kykene erottamaan syy-seuraussuhdetta näiden välillä. Kun tuloksia raportoidaan liiketoiminnan omistajalle, tulee raportoida myös piirteet, jotka ovat tuloksiin vaikuttaneet.

6. Tulosten evaluointi ja käyttöönotto

Tiedonlouhinnan jälkeen seuraavina vaiheina ovat saatujen tulosten evaluointi sekä tuloksista muodostetun tietämyksen käyttöönotto. Tulosten evaluoinnin tavoitteena on katsoa tuloksia liiketoimintatavoitteiden näkökulmasta. Tuottiko tiedonlouhinta tuloksia, joiden avulla on mahdollista saavuttaa tiedonlouhintaprosessin alussa asetetut tavoitteet? Evaluoinnin tarkoituksena on myös tarkistaa löydettyjen tulosten oikeellisuus. Jos esimerkiksi tiedonlouhinnan mallit on muodostettu opetusjoukon avulla, voidaan niiden oikeellisuus tarkistaa käyttämällä erillistä testausjoukkoa, jonka avulla mallien toimintaa testataan.

Evaluoinnin jälkeen seuraavana vaiheena on ottaa saadut tulokset käyttöön. Tässä vaiheessa on tarpeen palata jälleen prosessin alkuvaiheille ja katsoa mitkä asetetut tavoitteet olivat ja verrata niitä tuloksiin. Tavoitteita voidaan tässä kohtaa myös tarkastella uudelleen, mikäli tiedonlouhinnasta muodostui tuloksia, jotka siihen aihetta antavat.

Tulosten käyttöönotto voi tarkoittaa esimerkiksi muutoksia sivuston rakenteeseen, muutoksia tuotevalikoimaan tai muutoksia sivuston mainontaan. Tu-

loksia ja kerättyä dataa voidaan myös käyttää muodostamaan uutta sisältöä sivustolle. Jos käyttäjiltä on esimerkiksi kerätty palautetta heidän ostamistaan tuotteista, voidaan nämä palautteet näyttää tuotetietojen yhteydessä (kuten muun muassa Amazon.com tekee).

Tiedonlouhintaprosessin ei ole pakko päättyä käyttöönottoon. Tietämyksen muodostaminen voidaan mieltää jatkuvaksi prosessiksi, joka tuottaa sähköiseen kauppaan inkrementaalisia parannuksia. Käyttöönotosta saatuja tuloksia voidaan seurata, kun datan keräämistä ja tiedonlouhintaa jatketaan myös käyttöönoton jälkeen.

7. Yhteenveto

Tässä tutkielmassa on käsitelty tietämyksen muodostamista sähköisessä kauppapaikassa käyttäen pohjana Chapmanin ja muiden [1999] kuvaamaa yleistä tiedonlouhintaprosessia. Tutkielmassa on käsitelty datan keräämistä, esiprosessointia sekä tiedonlouhintaa ja louhinnasta saatujen tulosten käyttöönottoa.

Tutkielman aihepiirin ulkopuolelle jäivät muun muassa olemassa olevat työkalut. Tiedonlouhintaan, raportointiin sekä visualisointiin on olemassa useita valmiita työkaluja. Osa kyseisistä työkaluista on suunnattu WWW-sivustojen käyttöön, osa yleisempään käyttöön. Esimerkkinä WWW-sivuille suunnatusta työkalusta voidaan ottaa Googlen tarjoama Google Analytics -palvelu, joka mahdollistaa sekä datan keräämisen että raporttien muodostamisen datasta.

Viiteluettelo

[Ansari *et al.*, 2001] Suhail Ansari, Ron Kohavi, Llew Mason and Zijian Zheng, Integrating E-Commerce and Data Mining: Architecture and Challenges. In: *Proc. of 2001 IEEE International Conference on Data Mining*, 27-34.

[Chapman *et al.*, 1999] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer and Rüdiger Wirth, CRISP-DM 1.0 Step-by-step data mining guide. www.spss.ch/upload/1107356429_CrispDM1.0.pdf. Checked: 20.12.2011.

[Eckersley, 2010] Peter Eckersley, How Unique Is Your Web Browser? In: *Proc. of the Privacy Enhancing Technologies Symposium, Lecture Notes in Computer Science*, Springer, 1-19.

[Hussain *et al.*, 2010] Tasawar Hussain, Sohail Asghar and Nayyer Masood, Web Usage Mining: A Survey on Preprocessing of Web Log File. In: *International Conference on Information and Emerging Technologies,* 1-6.

[KDD Cup, 2000] http://www.kdd.org/kddcup/. Checked: 20.12.2011.

[Kohavi *et al.*, 2000] Ron Kohavi, Carla Brodley, Brian Frasca, Llew Mason, and Zijian Zheng. KDD-Cup 2000 organizers' report: Peeling the onion. In: *SIGKDD Explorations*, 2(2), (2000), 86-98.

[Kohavi, 2001] Ron Kohavi, Mining e-commerce data: the good, the bad, and the ugly, In: *Proc. of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 8–13.

[Kohavi *et al.*, 2004] Ron Kohavi, Llew Mason, Rajesh Parekh and Zijian Zheng, Lessons and Challenges from Mining Retail E-Commerce Data. *Machine Learning* **57** *(2004),* 83-113.

[Liu, 2011] Bing Liu, Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data. In: Bing Liu, Bamshad Mobasher and Olfa Nasraoui (eds.), *Web Usage Mining*, Springer Berlin Heidelberg, 2011, 527-594.

[McCarthy, 2000] Kieren McCarthy, Amazon makes regular customers pay more. http://www.theregister.co.uk/2000/09/06/amazon_makes_regular_customers_pay/. Checked: 20.12.2011.

[Tilastokeskus, 2010] Suomen virallinen tilasto (SVT): Tietotekniikan käyttö yrityksissä [verkkojulkaisu]. ISSN=1797-2957. Helsinki: Tilastokeskus. Saantitapa: http://www.stat.fi/til/icte/2010/index.html. Tarkistettu: 20.12.2011.

[YahooFinance, 2011] http://finance.yahoo.com/q/pr?s=AMZN+Profile. Checked 20.12.2011.

Elekäyttöliittymät: eleiden tunnistus ja vuorovaikutuksen suunnittelu

Joni Karvinen

Tiivistelmä.
Elevuorovaikutusta käytetään osana kosketuskäyttöliittymiä ja päällepuettavia sekä kontaktittomia elekäyttöliittymiä. Tässä tutkielmassa käsitellään nimenomaan kontaktittomien ja kehollisten elekäyttöliittymien eletunnistusmenetelmiä ja vuorovaikutuksen suunnittelua. Pääpaino on käsielein ja koko kehon elein ohjattavissa käyttöliittymissä.

Avainsanat ja -sanonnat: elekäyttöliittymät, eleiden tunnistus, vuorovaikutuksen suunnittelu, suunnittelumenetelmät
CR-luokat: H.1.2., H.5.2.

1. Johdanto

Elekäyttöliittymät eivät ole aivan uusi keksintö. Itse asiassa aikaisimpana esimerkkinä elekäyttöliittymistä voidaan käyttää venäläisen Leon Thereminin 1900-luvun alussa kehittämää elektronista soitinta, thereminia [Billinghurst, 2011]. Eletunnistuksen akateeminen tutkimus ja yritysten tutkimustyö aloitettiin jo 1960-luvulla, ja ensimmäiset kaupalliset elekäyttöliittymät tulivat markkinoille 1970-luvun lopussa [Myers, 1998]. Siitä huolimatta, että eleiden tunnistuksen tutkiminen on jatkunut jo muutaman vuosikymmenen ajan, vasta 2000-luvun puolella on todella siirrytty tutkimuslaboratorioista kuluttajille suunnattujen tuotteiden kehittelyyn.

Aiheen käsittely aloitetaan eleiden luokittelulla luvussa 2 ja elekäyttöliittymien luokittelulla luvussa 3. Tämän jälkeen käydään läpi eleiden tunnistuksessa käytettäviä menetelmiä ja luvussa 5 esitellään lyhyesti muutamia elekäyttöliittymiä eri sovellusalueilta. Luvussa 6 käsitellään vuorovaikutuksen suunnittelua. Asiaa lähestytään käytettävyyteen liitettyjen kriteerien kautta ja esittelemällä myös muutama suunnittelumenetelmä. Luku 7 on varattu loppupohdinnalle.

2. Eleiden luokittelu

Acharyan ja Mitran [2007] mukaan eleet ovat ilmaisevia sormien, käsien, pään ja kasvojen tai koko kehon liikkeitä, joilla on tarkoitus välittää merkityksellistä tietoa ja vuorovaikuttaa ympäristön kanssa. Lisäksi katseen suuntaaminen

voidaan mieltää eleeksi [Ansari et al., 2002].

Eleet ovat myös läheisessä yhteydessä puheeseen, sillä ne toimivat puhetta täydentävänä tai sitä tulkitsevana osana [Billinghurst, 2011]. Osaa käyttämistämme eleistä ei voi edes tulkita ilman niihin liittyvää puhetta niiden kontekstija kulttuuririippuvaisten ominaisuuksiensa takia.

Eleiden luokittelusta on kirjallisuudessa esitetty monia erilaisia vaihtoehtoja ja lähestymistapoja. Erilaisia luokitteluita ovat laatineet monien muiden joukossa muun muassa Rime ja Schiaratura, McNeill sekä Cadoz [Billinghurst, 2011]. Rime ja Schiaratura ovat päätyneet neljään luokkaan: symbolisiin, osoittaviin, ikonisiin ja pantomiimisiin eleisiin. McNeill lisää näiden kategorioiden rinnalle beat-eleet, joihin keskitytään elehdinnän käsittelemisen yhteydessä. Cadoz puolestaan jakaa eleet kolmeen kategoriaan niiden funktioiden mukaan: merkityksellistä tietoa välittäviin semioottisiin eleisiin, manipuloiviin ergoottisiin eleisiin ja epistemologisiin eleisiin, joita käytetään tiedon hankkimiseen ympäristöstä kosketuksen avulla.

Tässä luvussa tukeudutaan kuitenkin Karamin ja Schraefelin [2005] esittämään jaotteluun, joka perustuu aikaisempaan eleiden taksonomioita esittelevään kirjallisuuteen. Karam ja Schraefel [2005] ovat päätyneet viiteen eri kategoriaan: osoittavat eleet, manipulatiiviset eleet, semaforiset eleet, elehdintä ja kielelliset eleet. Tässä yhteydessä kiinnostuksen kohteena on tarkastella eleitä nimenomaan kone-ihminen-vuorovaikutuksen kontekstissa. Seuraavaksi käydään nämä kategoriat tarkemmin läpi ja niitä täydennetään myös muualta kirjallisuudesta löytyvällä materiaalilla.

2.1. Osoittavat eleet

Deiktiset eli osoittavat eleet ovat ihmiselle perustavanlaatuisimpia eleitä [Wexelblat, 1998]. Deiktiset eleet tarkoittavat osoittavia eleitä, joilla epäsuorasti pyritään identifioimaan kohde tai osoittamaan sen spatiaalinen sijainti [Karam and Schraefel, 2005]. Osoittavilla eleillä käyttäjä voi muun muassa valita elementtejä tai liikutella objekteja käyttöliittymässä joko kädellään osoittaen tai erityisen välineen, kuten kynän, avulla.

2.2. Manipulatiiviset ja semaforiset eleet

Manipulatiivisilla eleillä kontrolloidaan suorasti jotakin kohdetta siten, että virtuaalinen objekti noudattaa kädenliikkeellä annettua komentoa [Ansari et al., 2002]. Perinteisillä työvälineillä, kuten hiirellä tai kynällä, tehtyä suoraa manipulaatiota (esim. raahaaminen tai klikkaaminen) ei lasketa varsinaisiksi eleiksi vaan käyttäjän on tehtävä manipulatiivinen ele, jonka kone sitten tulkitsee komennoksi (esim. klikkaa ensin kohdetta ja tämän jälkeen klikkaa paikkaa, jonne haluaa objektin liikkuvan) [Karam and Schraefel, 2005].

Toisin kuin manipulatiiviset eleet, semaforiset eleet ovat ennaltamäärättyjä, symbolisia eleitä, joilla on kommunikatiivisia ominaisuuksia eli niiden välittämiä viestejä voidaan ymmärtää myös ilman puhetta [Ansari et al., 2002; Billinghurst, 2011]. Kommunikatiivisten piirteidensä lisäksi semaforiset eleet voidaan jakaa vielä staattisiin ja dynaamisiin eleisiin toisin kuin manipulatiiviset eleet, jotka ovat lähinnä dynaamisia [Karam and Schraefel, 2005]. Elekäyttöliittymissä semaforisia eleitä on laajalti käytetty, vaikkakin tällaisten eleiden käyttöä on pidetty epäluontevana, koska eleet ovat epäaitoja ja opeteltavia eivätkä siksi intuitiivisia käyttää [Wexelblat, 1998; Ansari et al., 2002].

2.3. Elehdintä

Elehdinnällä *(gesticulation)* viitataan puheen rinnalla tehtäviin vapaamuotoisiin eleisiin [Ansari et al., 2002], joilla pyritään välittämään informaatiota puheen kohteen koosta, muodosta tai suunnasta [Billinghurst, 2011]. Samaa asiaa tarkoittaen kirjallisuudessa käytetään myös käsitteitä ikoniset tai havainnolliset eleet [Karam and Schraefel, 2005].

Elehdintään liittyviksi voidaan lukea myös jako edelleen pantomiimisiin eleisiin, embleemeihin ja beat-eleisiin sekä koheesiivisiin eleisiin. Pantomiimieleet ovat tyypillisesti eleitä, joilla puhuja kuvaa jotakin toimintaa jäljittelemällä sitä eleillään [Billinghurst, 2011]. Embleemit ovat ilmaisuja, joilla on jokin ennalta-asetettu merkitys, esimerkiksi peukalot ylhäällä merkitsee, että kaikki on hyvin [Ansari et al., 2002].

McNeill [1992] lisää osoittavien, ikonisten ja pantomiimisten eleiden taksonomiaan vielä beat-eleet ja koheesiiviset eleet. Beat-eleitä tehdään puheen rytmin tahdissa ja tyypillisesti ne ovat yksikertaisia eleitä, kuten sormien naputus, eikä niillä pyritä sinänsä välittämään tietoa tai viestiä. Koheesiiviset eleet ovat kerrontaa koossapitäviä eleitä, joilla puhuja pyrkii pitämään ajallisesti erillään olevia, mutta temaattisesti samanlaisia osia yhdessä. Tällaiset koossapitävät eleet voivat olla mihin tahansa muuhun kategoriaan kuuluvia eleitä, mutta koheesiivisilla eleillä painotetaan jatkuvuutta puheessa [McNeill, 1992].

2.4. Kielelliset eleet

Kielellisten eleiden, kuten viittomakielen, voidaan katsoa kuuluvan semaforisten eleiden kategoriaan, sillä nämä eleet ovat ennalta sovittuja ja opeteltuja, mutta erona on kuitenkin viittomakielen kieliopilliset komponentit ja tarkoitus pyrkiä vastavuoroiseen kommunikaatioon, jolloin viittomakieltä ei voida liittää myöskään elehdintäkategoriaan. Haasteeksi ihminen-kone-vuorovaikutuksessa koituu koneen kyky yhdistellä monien eleiden ketjuja ja ymmärtää monimutkaisempia lauserakenteita kuin pelkästään yksittäisten ja staattisten merkkien

tulkitsemista. [Karam and Schraefel, 2005]

3. Elekäyttöliittymien luokittelu

Karam ja Schraefel [2005] jakavat elekäyttöliittymätyypit kahteen pääkategoriaan: havainnollisiin *(perceptual)* ja ei-havainnollisiin *(non-perceptual)*. Näitä kahta luokkaa voidaan edelleen jakaa osiin erilaisten elevuorovaikutuksen mahdollistavien teknologioiden mukaan.

Havainnollisilla käyttöliittymillä tarkoitetaan kontaktittomia käyttöliittymiä. Ajatus käyttöliittymien nimeämisestä havainnollisiksi on, että kone pystyy havainnoimaan käyttäjän liikkeitä ja sijaintia tilassa tai audiosensorien avulla ymmärtämään käyttäjän puhekomentoja ilman, että käyttäjän tarvitsee huolehtia lisälaitteiden käytöstä [Karam ja Schraefel, 2005]. Tällaiset käyttöliittymät toteutetaan konenäkömenetelmien tai etäsensoritekniikan avulla [Karam ja Schraefel, 2005].

Ei-havainnolliset käyttöliittymät vaativat fyysisen kosketuksen tai erityisiä laitteita niiden käyttämiseen. Fyysinen kontakti voidaan tehdä kosketuksena tai esimerkiksi kynällä. Erityisiä lisälaitteita ovat muun muassa elektronista sensoritekniikkaa käyttävät laitteet, kuten datahansikkaat. Ei-havainnollisten käyttöliittymien joukkoon Karam ja Schraefel [2005] sijoittavat myös käsinkosketeltavat käyttöliittymät *(tangible interface)*, joilla fyysistä objektia voidaan manipuloida tietokoneen ruudulla. Erillisiä laitteita hyödynnetään edelleen, vaikka niiden käyttöä yleisesti pidetään kömpelönä ja rasittavana sekä esteenä mahdollisimman luonnolliselle ja vapaalle vuorovaikutukselle. Lisäksi Karam ja Schraefel [2005] sijoittavat ei-havainnollisten kategoriaan käyttöliittymät, jotka hyödyntävät koputusmaisia tai naputusmaisia äänisyötteitä kohteen paikantamiseksi. Tämä luokittelu perustuu siihen, että käyttäminen vaatii yhä fyysisen kosketuksen eikä äänisyöte ole käyttäjältä tarkoituksellista.

4. Eleiden tunnistuksessa käytettäviä menetelmiä

Acharya ja Mitra [2007] jakavat eleiden tunnistuksessa käytettävät menetelmät tilastollisen mallintamiseen menetelmiin, konenäön hahmontunnistus- ja kuvanprosessointitekniikoihin ja soft computing -menetelmiin sekä konnektionistisiin lähestymistapoihin. Koska erilaisia tekniikoita on kehitetty suuri määrä, esitellään eri osa-alueilta käytetyimpiä ja suosituimpia lähestymistapoja.

4.1. Tilastollinen mallintaminen

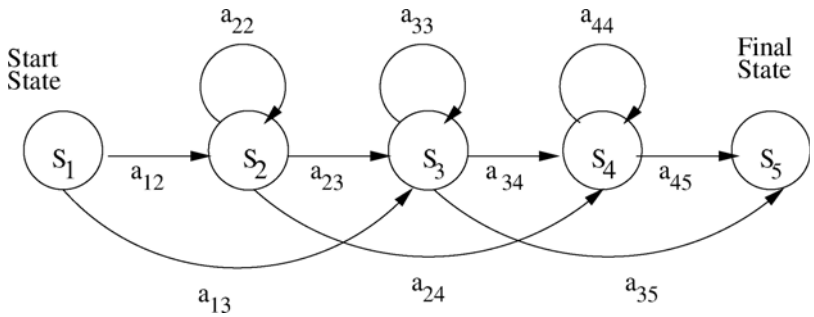
Tilastollisista menetelmistä esitellään yleisellä tasolla piilotettu Markovin malli ja äärellinen automaatti. Kummankin menetelmän kohdalla esimerkkinä käytetään kehon eleiden tunnistamista.

4.1.1. Piilotettu Markovin malli

Piilotettu Markovin malli *(Hidden Markov Model)* on yksi käytetyimmistä tilastollisista menetelmistä eleiden tunnistuksessa. Malli on matemaattiselta rakenteeltaan rikas ja toimii tehokkaasti käytännön sovelluksissa, kuten puheen- ja eleentunnistuksessa sekä bioinformatiikan alan sovelluksissa, joissa tarvitaan spatiotemporaalisen informaation käsittelyä [Acharya and Mitra, 2007]. Temporaalisen komponentin ansiosta Markovin piilomalli soveltuu erityisesti dynaamisten eleiden tunnistamiseen [Acharya and Mitra, 2007].

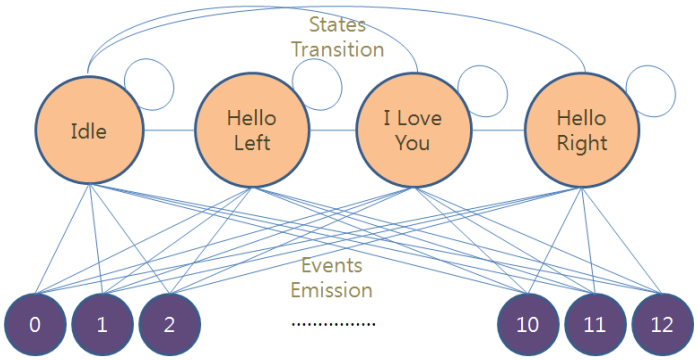
Piilotettu Markovin malli on kaksoisstokastinen prosessi, jonka perustana on Markovin ketju, joka sisältää äärellisen määrän eri tiloja ja joukon yhteen tilaan liittyviä satunnaisia funktioita [Acharya and Mitra, 2007]. Siirtyminen eri tilojen välillä riippuu tilojen siirtymätodennäköisyyksistä *(state transition probability)* ja tilan tuottama tulos riippuu sen havaintotodennäköisyydestä *(output probability)* [Acharya and Mitra, 2007]. Lisäksi mallille määritetään alkutilan todennäköisyys *(initial state probability)* [Ishii et al., 1992; Acharya and Mitra, 2007]. Tietty tila millä tahansa ajanhetkellä riippuu ainoastaan sitä edeltävästä tilasta ja siirtymäpolut voivat osoittaa myös tilaan itseensä, jolloin tietyssä tilassa voidaan pysyä kuinka kauan tahansa [Ishii et al., 1992]. Dynaamisten eleiden tunnistuksessa käytetään ajassa vasemmalta oikealla etenevää topologiaa, jossa tila voi palata joko itseensä tai siirtyä seuraaviin tiloihin (ks. kuva 1) [Acharya and Mitra, 2007]. Tiukempi malli on LRB-malli *(left-right-banded)*, jossa tila voi siirtyä ainoastaan sitä seuraavaan tai palata itseensä [Acharya and Mitra, 2007]. Mallin nimi perustuu siihen, että tiedetään ainoastaan havaintojen muodostamat sekvenssit, mutta tiloja itsessään ei havaita [Ishii et al., 1992; Acharya and Mitra, 2007].

Piilotetun Markovin mallin vahvuus on sen oppimiskyky, joka saavutetaan siten, että automaattisesti optimoidaan mallin siirtymä- ja havaintotodennäköisyyksiä niiden maksimien löytämiseksi [Ishii et al., 1992]. Mallin opettamiseen käytetään muun muassa Baum-Welch- ja Viterbi-algoritmeja [Acharya and Mitra, 2007].



Kuva 1. Vasemmalta oikealle etenevä topologia. [Acharya and Mitra, 2007]

Islam ja muut [2011] ovat käyttäneet piilotettua Markovin mallia ylävartalon eleiden tunnistuksessa. Kuvassa 2 on esitettynä neljän eri eleen tunnistamisessa käytetyn mallin rakenne. ”Idle”-ele on neutraali asento, jossa henkilö pitää kädet alhaalla. ”Hello left” on tervehdys vasemmalla kädellä ja ”Hello right” oikealla kädellä. ”I love you” -eleessä henkilö nostaa molemmat kätensä päänsä päälle. Jokainen ele on siis oma tilansa. Eleentunnistuksessa käytetään 13 avainasentoa, jotka tallennetaan omaan kirjastoonsa *(key pose library, KPL).* Kirjasto on itse asiassa vektori, joka sisältää jokaisen eri avainasennon. Avainasento puolestaan koostuu parametriarvoista, joilla määritellään kuvasta ylävartalon osat ja tiettyä asentoa esittävästä silhuettikuvasta. Kun koneelle annetaan syötteenä silhuettikuva henkilöstä, käyttämällä avainasennosta tallennettuja arvoja kone pystyy laskemaan, mitä avainasentoa lähimpänä käyttäjän asento on. Asennoista muodostetut sekvenssit toimivat mallin havaintoina ja tilojen siirtymä- sekä havaintotodennäköisyydet vastaavat kunkin syötekuvan kohdalla laskettuja painotuksia.



Kuva 2. Tilojen ja havaintojen väliset suhteet. [Islam et al., 2011]

4.1.2. Äärellinen automaatti

Äärellisessä automaatissa *(finite state machine, FSM)* tietty ele mallinnetaan järjestetyksi sarjaksi erilaisia tiloja [Acharya and Mitra, 2007]. Eleestä tallennetaan mahdollisimman monta prototyyppiä eli esimerkillisiä vaiheita tai liikeratoja tietyn eleen tuottamisessa. Vaiheesta toiseen siirrytään silloin, kun syötettävä data eli käyttäjän liike vastaa prototyypin vaihetta, ja kun saavutetaan viimeinen vaihe, ele tulkitaan tunnistetuksi [Acharya and Mitra, 2007]. Spatio-temporaalisen luonteensa ansiosta äärellisen automaatin käyttö soveltuu dynaamisten eleiden tunnistukseen.

Esimerkkinä äärellisen automaatin käytöstä esitellään lyhyesti Chanin ja muiden [2011] luoma interaktiivinen tanssipeli, jossa pelaaja opettelee go-go-tanssia virtuaalisen partnerin kanssa. Ideana on, että virtuaalinen tanssiopettaja ja -kumppani opettaa ensin tietyn tanssiliikkeen, jonka jälkeen pelaajan on toistettava tämä tanssiliike. Erilaisia liikkeitä on yhteensä kahdeksan ja niistä jokaisesta on talletettu oma mallinsa, johon pelaajan liikettä verrataan.

Pelissä käytettävä automaatti koostuu neljästä eri vaiheesta: toimettomasta tilasta *(idle state)* eli tilasta, jolloin datasyöte on vielä tuntematon, aloitustilasta, responssitilasta ja päätöstilasta. Aloitustilassa malli jaetaan viiten lyhyempään osaan (ks. kuva 3). Alussa tunnistetaan, vastaako pelaajan tanssiliike mitään kahdeksasta malliliikkeestä vertaamalla aloitusasentoja. Liikkeen tunnistamiseksi on määritelty ennalta tiettyjä laskennallisia raja-arvoja, joita ei saa ylittää. Jos kynnysraja ylittyy, liike ei vastaa mallia ja tällöin palataan aloitustilasta takaisin toimettomaan tilaan.

Kuva 3. Kuvassa esitettynä yhden tanssiliikkeen vaiheet. [Chan et al., 2011]

Kun tietty tanssiliike on tunnistettu, responssitilassa seurataan, jatkaako pelaaja tanssiliikettä mallin mukaisesti. Järjestelmä laskee prosentuaalisesti, kuinka tarkasti henkilö pystyy toistamaan liikkeen ja jos virheellisyydelle sallittu kynnysarvo ylittyy, palataan jälleen toimettomaan tilaan. Päätöstila saavutetaan, kun tanssiliike on pystytty toistamaan 100 %:n tarkkuudella.

4.2. Konenäkö, hahmontunnistus ja kuvanprosessointitekniikat

Verrattuna päällepuettaviin tai muulla tavalla laitteistoa vaativiin käyttöliittymiin konenäköä hyödyntävät sovellukset ovat monella tapaa etulyöntiasemassa. Konenäkötekniikan käyttäminen vapauttaa käyttäjän "käyttöliittymän asennustyöstä" eikä tekniikka rajoitu tunnistamaan ainoastaan tietynlaisia eleitä [Edan et al., 2011]. Kuluttajalle laitteisto on lisäksi suhteellisen edullista, mutta toisaalta algoritmit sekä ohjelmointityö ovat sovelluskohtaisia ja eleiden tunnistamista vaikeuttavat monet eri huomioonotettavat asiat, kuten taustan yksityiskohdat ja ympäristön valaistus [Edan et al., 2011]. Erilaisia konenäköön liittyviä ongelmia on pyritty ratkaisemaan hahmontunnistus- ja kuvanprosessointitekniikoiden avulla.

4.2.1. Piirreirrotus

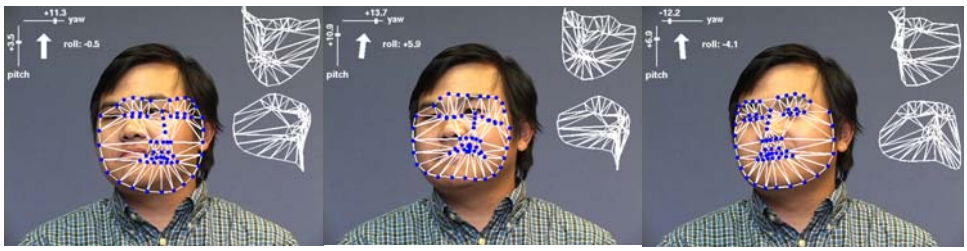
Piirreirrotuksella *(feature extraction)* tarkoitetaan prosessia, jossa määritetään joukko piirteitä, joiden avulla tunnistus voidaan tehokkaasti suorittaa. Piirreirrotuksen toteutukselle on olemassa suuri määrä erilaisia toteutustapoja. Eräitä tunnistustapoja ovat muun muassa värin perusteella tehtävä piirteiden erottaminen tai muodon perusteella tehtävä tunnistus.

Varsinkin pään ja käsien alueet voidaan tunnistaa kohtuullisen tarkasti

pelkästään värin perusteella [Edan et al., 2011]. Ihonväri irrotetaan taustastaan ja pystytään segmentoimaan erilaisten ominaisuuksien perusteella, kuten värikylläisyyden ja saturaation avulla [Edan et al., 2011]. Menetelmä ei välttämättä toimi tehokkaasti, mikäli tausta on samanvärinen kuin tunnistettava kehon osakin, mutta tunnistuksen helpottamiseksi on käytetty markkereita, esimerkiksi väritettyjä hansikkaita, joiden mukaan segmentointi voidaan tehdä [Edan et al., 2011]. Tämänkaltaiset ratkaisut kuitenkin heikentävät käytettävyyttä, koska käyttäjän tarvitsee käyttää ylimääräisiä varusteita.

Eleiden tunnistusta voidaan tehdä myös objektin muodon perusteella. Cootes ja Taylor [1992] ovat kehittäneet Active Shape Model -menetelmiä *(ASM, kutsutaan myös Smart Snakes -nimellä)*, jotka iteratiivisesti parantavat asennon, skaalan ja muodon estimaatioita kuvisssa. Jokainen kuva ymmärretään joukkona pisteitä ja pistejakaumamallin avulla *(point distribution model)* jokainen piste pyritään siirtämään parempaan positioon laskettujen arvioiden perusteella.

ASM-mallit liittyvät läheisesti AAM-malleihin *(Active Appearance Model).* Usein kasvojen tunnistamisessa käytetyt AAM-mallit ovat Bakerin ja Matthewsin [2004] määritelmän mukaan generatiivisia malleja tietystä visuaalisesta ilmiöstä. AAM:n sovitus kuvaan koostuu syötekuvan ja lähimmän malli-instanssin välisen virheen minimoimisesta (ks. kuva 4).



Kuva 4. Vasemmalla AAM-mallin sovituksen alkuvaihe. Keskellä 30 iteraation jälkeen ja oikealla viimeinen iteraatiovaihe. [Baker et al., 2004b]

AAM-mallin pisteverkon sijainnin määrityksessä voidaan käyttää esimerkiksi optical flow -algoritmia [Baker et al., 2004a]. Optical flow -menetelmällä lasketaan arvio kuvan pisteiden liikkeestä. Menetelmän hyötynä on, ettei tarvitse irrottaa ja laskea estimaatiota jokaiselle pisteelle erikseen vaan liikeala *(motion field)* perustuu approksimoinnille. Optical flow -menetelmä toimii hyvin, jos liike on hidasta eikä liikkuvia osia ole kovin monta, joten kyseinen tapa sopii kasvojen tunnistamiseen, koska kasvojen alueet (kuten otsan tai leuan alueet) muodostavat hienojakoisia tekstuureja. [Acharya and Mitra, 2007]

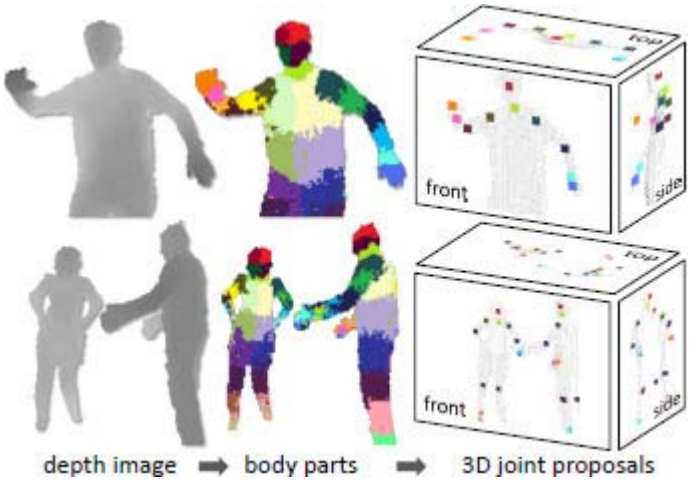
Piirreirrotuksesta saatua dataa käytetään lopulta erilaisten tilastollisten menetelmien syötteenä. Lopullinen eleen tunnistaminen voidaan tehdä erilaisten luokittelijoiden *(classifier)* avulla, joista piirreirrotuksen yhteydessä on

käytetty muun muassa aiemmin esiteltyä piilotettua Markovin mallia.

4.2.2 Syvyyskuvat

Syvyysinformaation hyödyntäminen on osoittautunut koko kehon eleiden tunnistamisessa tehokkaaksi tavaksi. Syvyyskameroiden ja -kuvien käytöllä on useita hyötyjä, sillä syvyyskuvat toimivat hämärässäkin, ne antavat kalibroidun skaalaestimaatin, eivät ota huomioon väriä tai tekstuuria ja selvittävät silhuettien tulkinnanvaraisuudet asennon tunnistamisessa. Syvyyskuvien avulla yksinkertaistetaan myös taustan irrotusta [Blake et al., 2011].

Microsoftin Kinect hyödyntää erityisen tehokkaasti syvyyskuvien käyttöä tunnistaessaan pelaajan kehon asentoja. Kun Kinect-sensori tunnistaa pelaajan, tästä luodaan ensin syvyyskuva (ks. kuva 5). Kuvan jokaisen pikselin kohdalla määritellään, mihin kehon osaan se kuuluu vai onko se kehon ulkopuolella. Tällä tavalla voidaan luoda kuvan 5 mukainen kehon osien segmentointi. Kinect tunnistaa kehon eri osat hyvin tarkasti, koska taustalla toimivaa satunnaista päätöspuuluokittelijaa on opetettu sadoilla tuhansilla erilaisilla kuvilla ihmisistä eri asennoissa ja tekemässä erilaisia liikkeitä. Viimeisessä vaiheessa algoritmi tuottaa pelaajasta kolme luurankomaista 3D-mallia *(3D joint proposal).* Kaikki tämä toimii erittäin tehokkaasti. Algoritmi käydään läpi 200 kertaa sekunnissa, joten reaaliaikainen liikkeentunnistus onnistuu helposti. [Blake et al., 2011]



Kuva 5. Kinectin tuottamat kuvat.

4.3. Soft computing ja konnektionistiset lähestymistavat

Soft computing -menetelmät tarjoavat joustavia ratkaisuja tilanteisiin, joissa vaaditaan sietokykyä epätarkkuudelle, epävarmuudelle, likimääräiselle päättelylle ja osittaiselle totuudelle [Acharya and Mitra, 2007].

Soft computing -menetelmiä ovat muun muassa sumeat ja karkeat joukot, keinotekoiset neuroverkot *(artificial neural networks, ANN)* ja geneettiset algoritmit *(genetic algorithm, GA).* Konnektionistisia lähestymistapoja ovat muun muassa neuroverkkomallit, kuten MLP *(multilayer perceptron),* TDNN *(timedelay neural network)* ja RBFN *(radial basis function network)* [Acharya and Mitra, 2007].

TDNN ja muut uusiutuvat neuroverkot toimivat hyvin dynaamisten käsieleiden tunnistamisessa, kun tarvitaan ajallisten ja kontekstuaalisten suhteiden huomioonottamista [Acharya and Mitra, 2007]. TDNN-mallia on käytetty menestyksekkäästi esimerkiksi tunnistamaan amerikkalaista viittomakieltä.

5. Elekäyttöliittymiä eri sovellusalueilta

Elepohjaisessa vuorovaikutuksessa piilee potentiaalia, jonka täydellä käyttöönotolla on varmasti useimpia eri sovellusalueita rikastuttavia vaikutuksia. Pelimaailman innovaatioita ollaan yhä enemmän tuomassa jokapäiväisempään käyttöön ja varsinkin Kinectin pioneerimaista teknologiaa hyödynnetään pelikehityksen ulkopuolellakin varsin paljon.

Aluksi tässä luvussa esitellään elekäyttöliittymiä arkipäiväisessä käytössä. Seuraavaksi tarkastellaan kahta musiikkisovellusta. Oppimisen ja koulutuksen yhteydessä esitellään uudenlaisten opetusohjelmien kehittämiseen keskittyvä projekti KinectEDucation ja esitellään Kinectilla toimiva matematiikan opetusohjelma. Tämän jälkeen käydään läpi esimerkkejä terveydenhuollon ja lääketieteen alalta. Viimeisenä lyhyesti tarkastellaan elevuorovaikutuksen hyötyjä kriisinhallintaan suunnitellussa käyttöliittymissä.

5.1. Viihdekäyttö ja arkipäiväinen elektroniikka

Lähinnä peli- ja mobiiliteollisuuden ansiosta jo miljoonat ihmiset käyttävät elepohjaista vuorovaikutusta hyödyntäviä tuotteita tai ovat ainakin saaneet jonkinlaisen vaikutelman elekäyttöliittymistä [Haywood et al., 2011]. Pelikonsolit, kuten Nintendo Wii, Microsoft Kinect ja Playstation Move ja kosketusnäytölliset laitteet ovat tuoneet elekäyttöliittymät suuren yleisön tietoisuuteen.

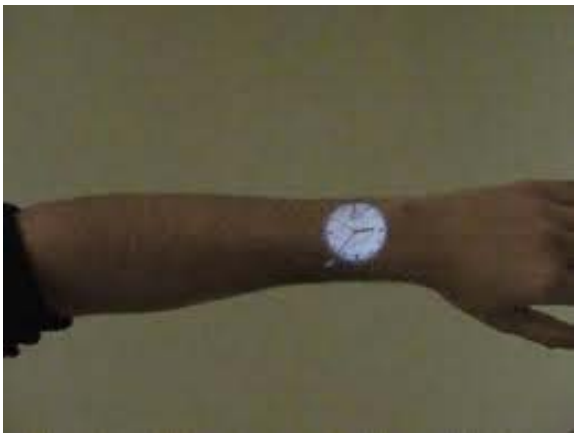
Useimmille käyttäjille oman kehon käyttäminen ohjaamaan käyttöliittymää on varmasti mielenkiintoista, mukaansatempaavaa ja paljon hauskempaa kuin hiirellä, näppäimistöllä tai ohjaimella pelaaminen. Hauskuus onkin merkittävä osa käyttökokemusta, mitä elekäyttöliittymät ovat pystyneet lisäämään. Erityisesti pelien tarjoama kokemus on vangitsevampi, sillä pelaaja pystyy oikeasti heilauttamaan kättään ikään kuin tällä olisi tennismaila kädessä tai miekkailemaan roolipelissä vihollisiaan vastaan. Samanlaisen kokemuksellisuuden ei tarvitse rajoittua pelkästään peleihin vaan sitä pitäisi hyödyntää myös arkipäiväisemmissä sovelluksissa.

Eräs esimerkki jokapäiväisen teknologian kokemuksellisuuden lisäämiseen

on Pranav Mistryn kehittämä SixthSense [Mistry, 2010]. SixthSense on vielä prototyyppivaiheessa oleva elekäyttöliittymä, joka laajentaa fyysistä maailmaa digitaalisella informaatiolla. Nimi SixthSense juontuu ajatuksesta, että kyseinen laite laajentaa lisätyn todellisuuden avulla tietämystämme ympäröivästä maailmasta toimien näin ikään kuin kuudentena aistinamme.

Vaikka SixthSense varsinaisesti voidaan mieltää päällepuettavaksi käyttöliittymäksi, ei sen käyttö kuitenkaan ole yhtä intrusiivista kuin esimerkiksi datahansikkaiden tai -kypärien käyttö, sillä tekniikka perustuu ihmisillä aina mukana kulkeviin laitteisiin. SixthSense koostuu taskuprojektorista, peilistä ja kamerasta, jotka yhdistetään käyttäjän mobiililaitteeseen. Projektorin projisoinnin avulla voidaan erilaisista pinnoista tai seinistä tehdä käyttöliittymän alustat, joten SixthSense mahdollistaa elekäyttöliittymän käytön missä tahansa käyttäjä ikinä liikkuukaan eikä rajoita tätä pysymään tietyssä paikassa. Konenäkömenetelmää käyttäen kamera tunnistaa käyttäjän kädenliikkeet. Laite tukee myös useamman käyttäjän liikkeiden tunnistamista. Vielä prototyyppivaiheessa on konenäköä varten käytetty eri värisiä markkereita sormissa, mutta tunnistuksen kehittyessä niitä ei varmastikaan vaadita lopullisessa tuotteessa.

Kuva 6. Pranav Mistry tekee "kuvanrajaus"-eleen. [Mistry, 2010]



Kuva 7. Käyttäjän ranteeseen heijastettava virtuaalinen kello. [Mistry, 2010]

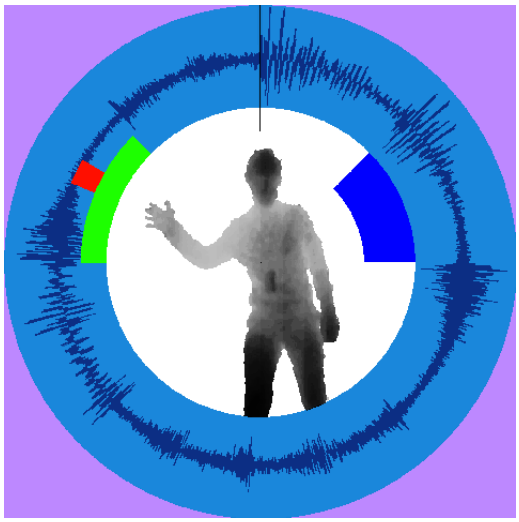
SixthSensen tunnistettavat eleet ovat mielikuvituksellisia. Esimerkiksi jos käyttäjä haluaa ottaa jostakin tilanteesta kuvan, hänen täytyy vain tehdä "kuvanrajaus"-ele eikä mitään muuta (ks. kuva 6). Ja kun käyttäjän tarvitsee tietää, mitä kello on, hän saa ranteeseensa virtuaalisen kellon piirtämällä ympyrän ilmassa (ks. kuva 7).

5.2. Musiikkisovelluksia

Muusikot ja muut taiteen tekijät saavat elekäyttöliittymästä uusia työvälineitä.

Visuaalisen taiteen tekijöille elevuorovaikutus tarjoaa mahdollisuuden luoda entistä interaktiivisempia teoksia ja etenkin elektronisen musiikin tekijöille uudet sovellukset vievät ilmaisun uusiin ulottuvuuksiin.

Kinect BeatWheel on musiikkitiedostojen "samplaamiseen" eli lyhyiden nauhoitteiden manipuloimiseen kehitetty käyttöliittymä [Music Hack Day, 2011]. "Luuppaava" eli koko ajan toistuva ääniraita on jaettu kahdeksaan osaan (ks. kuva 8). Yksi osa vihreän palikan kokoinen alue ja vihreä palikka osoittaa osan, jota parhaillaan toistetaan. Punainen palikka osoittaa kohdan, johon käyttäjän käsiele kullakin hetkellä osoittaa. Sininen palikka on tempoindikaattori ja se liikkuu kehää ympäri musiikin tahdissa. Heilauttamalla kättään käyttäjä pystyy valitsemaan, mitä osaa "samplesta" kullakin hetkellä toistetaan. Lisäksi käyttäjä voi toistaa äänitiedostoa joko etu- tai takaperin. Kun käyttäjä heilauttaa kättään vastapäivään, "sample" soitetaan takaperin.



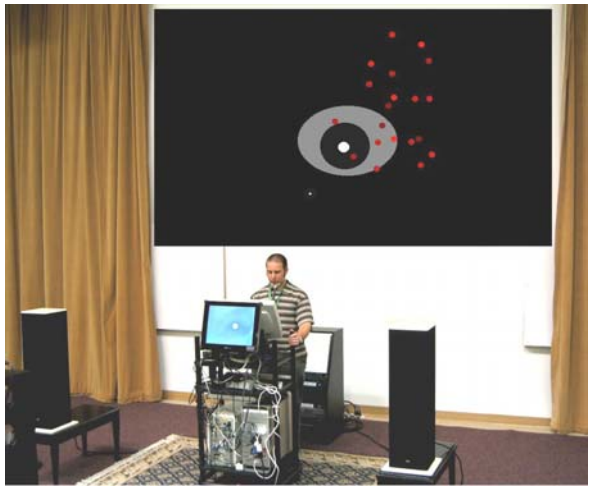
Kuva 8. Kinect BeatWheelin käyttöliittymä. [Music Hack Day, 2011]

Yllättävämmän lähestymistavan musiikin tekemiseen esittelevät Halverson ja muut [2008] EyeMusic-sovelluksellaan, jolla voi silmänliikkeillä säveltää ja esittää musiikkia. Ideana on, että sävellyksestä tehdään visuaalinen esitys ja esitettäessä teosta erilaiset visuaaliset objektit tai (x,y)-koordinaatit vastaavat ennaltaäänitettyjä ääniä. Äänet voidaan ohjelmoida vastaamaan tiettyjä soittimia ja edelleen tiettyjä säveliä tai ne voivat olla myös muita kuin perinteisten instrumenttien ääniä. Ääniä toistetaan kohdistamalla katse tiettyyn visuaaliseen kohteeseen.

Kuvan 9 tilanteessa esitetään musiikkiteosta, jossa katseen liikuttaminen tuottaa rohisevan ja tärisevän äänimaailman. Esittäjä pystyy manipuloimaan stereoääntä silmänliikkeillään siten, että siirtämällä katsetta oikealle tai vasemmalle tämä voi muuttaa äänen panorointia. Kun katse kohdistetaan pieneen punaiseen täplään, tuotetaan eräänlainen piippausääni. Lisäksi räpyttämällä silmiä tuotetaan voimakas läimäysääni.

Esitetty teos ei ole ainoa, jonka Halverson ja muut [2008] muusikoiden kanssa toteuttivat. Eräs teos oli toteutettu siten, että maalaus jaettin (x,y)-pisteisiin ja jokaista koordinaattia vastasi jokin ääni. Lisäksi ryhmä toteutti virtuaalisen pianon, mutta rytmin kontrollointi tuotti vaikeuksia kokeneellekin muusikolle vielä useiden viikkojen harjoittelun jälkeen.

Halverson ja muut [2008] näkevät silmänliikkeillä kontrolloidun sovelluksen kehittämisellä monia hyötyjä. Ensiksikin säveltäminen ja musiikin esittäminen viedään uudelle, kiehtovalle alueelle ja käsitys perinteisistä soittimista laajenee. Silmänliikkein kontrolloidulla sovelluksella myös suodaan mahdollisuus musiikilliseen ilmaisuun niille, joilla on motorisia vaikeuksia tai muita fyysisiä esteitä eivätkä täten kykene soittamaan mitään instrumenttia. Lisäksi tutkimalla silmänliikkeiden musiikillisia ominaisuuksia saadaan tuotettua uudenlaista data-aineistoa, josta saattaa jatkossa olla merkittävää hyötyä muun muassa tietojenkäsittelytieteelle tai kognitiiviselle psykologialle, jotka eye tracking -tutkimusta tarvitsevat.



Kuva 9. Kuvassa esitetään yhtä osaa kokonaisesta musiikkiteoksesta. Keskellä ruutua on piirros silmästä, joka toistaa esittäjän silmänliikkeitä ja katseen suuntaamista. Silmän alapuolella oleva pieni valkoinen piste osoittaa kohdan, johon esittäjä sillä hetkellä katsoo.

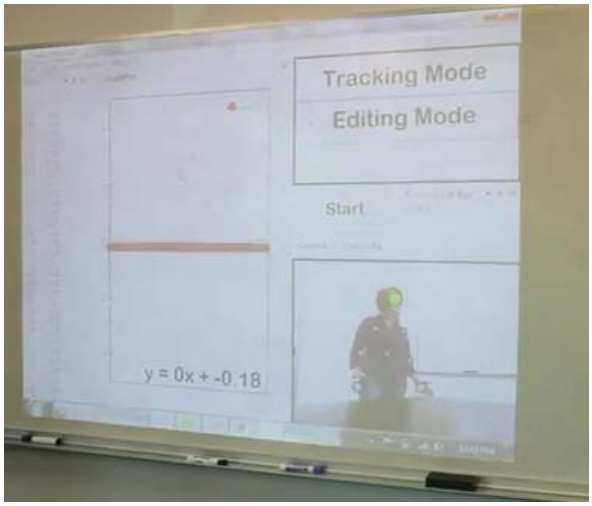
5.3. Oppiminen ja koulutus

Perinteisiä opetustilanteita on pitkään pidetty liian passiivisina ja vuorovaikutukseltaan yksipuolisina. Oppilaat kaipaavat kiinnostavaa ja mielekästä tekemistä tunneille sen sijaan, että istuisivat aloillaan ja olisivat pelkästään kuuntelijan roolissa. Elevuorovaikutusta hyödyntävien opetusohjelmien avulla oppilaasta voidaan tehdä aktiivisempi toimija. Haywood ja muut [2011] näkevät, että elepohjaista vuorovaikutusta käyttävillä ohjelmilla parannetaan entisestään yhteisöllisen oppimisen keinoja, sillä uuden tekniikan avulla kokonainen ryhmä voi yhdessä selvittää opittavaa asiaa visualisointien ja

simulaatioiden avulla. Esimerkiksi Kinect sallii kuusi yhtäaikaista käyttäjää, joten pienryhmissä voitaisiin opiskella vaikeiksi koettuja fysiikan ilmiöitä näitä simuloimalla tai tutkia ihmisen aivojen rakenteita 3D-mallia käsittelemällä.

Juuri Kinect-pohjaisten opetusohjelmien kehitykseen on perustettu KinectEDucation-yhteisö. Sen ajatuksena on vapaasti jakaa ja kehittää opetusohjelmia sekä ideoida, miten muun muassa hyödyntää jo olemassa olevia pelejä opetuksessa [KinectEDucation, 2011].

Yksi KinectEDucationin kautta jaettava opetusohjelma on Kinect Math (ks. kuva 10), jonka kehittäjien ajatuksena oli tehdä abstraktista matematiikan opiskelusta visualisointien avulla paljon havainnollisempaa ja konkreettisempaa. Ohjelmassa on kolme eri moduulia: seuranta- *(tracking mode)*, muokkaus- *(editing mode)* ja sovittamismoduuli *(matching mode)*. Seurantamoduulissa oppilaan tehtävänä on liikkua kameran edessä paikasta A paikkaan B ja ohjelma mittaa tämän nopeuden ja kiihtyvyyden sekä piirtää tilanteesta kuvaavan grafiikan. Muokkausmoduulissa oppilaan on mahdollista manipuloida käsien liikkeillä kuvaajia ja ohjelma näyttää kuvaajan yhtälön. Sovittamismoduulissa ohjelma satunnaisesti esittää jonkin kuvaajan ja oppilaan tehtävänä on liikkua sensorin edessä siten, että tämän omasta liikkumisesta voitaisiin piirtää kyseinen kuvaaja.



Kuva 10. Oikealla alhaalla videokuvaa käyttäjästä ja vasemmalla alue, johon kuvaajat piirretään. [Kuvalähde: http://channel9.msdn.com/coding4fun/kinect/Reaching-out-and-touching-Math-with-Kinect-Math]

5.4. Terveydenhuolto ja lääketiede

Uudet pelikonsolit ovat antaneet panoksensa myös terveydenhuoltoon. Esimerkiksi Nintendo Wiitä on menestyksekkäästi käytetty Parkinsonin tautia sairastavien potilaiden kuntouksessa, niin kutsutussa Wii-habissa. Kaksikymmentä Parkinson-potilasta pelasi Wiin tennis-, keilaus- ja nyrkkeilypelejä kolme kertaa viikossa neljän viikon ajan. Jo tässä ajassa potilaiden motorisissa

taidoissa oli havaittavissa merkittäviä positiivisia muutoksia ja joidenkin pelaajien masennus oli merkittävästi lieventynyt [LiveScience, 2009].

Koska esimerkiksi Wii-pelit voivat olla liian vaikeita käytettäväksi kuntoutuksessa eikä niiden kehonliikkeiden tunnistus ole riittävän tarkkaa, Bolas ja muut [2011] toteuttivat halvauspotilaiden kuntoutukseen tarkoitetun pelin. Peli sijoittuu jalokivikaivokseen ja pelaajan tavoitteena on kerätä ruudulla olevat kahdeksan jalokiveä ohjaamalla avatar-hahmoaan (ks. kuva 11). Peli on suoraviivaisen yksinkertainen, koska päätavoitteena on mukailla terapeuttisia liikkeitä.

Pelaajan liikkeiden tunnistamisessa käytettiin samaa teknologiaa kuin Microsoftin Kinectissa. Ohjelmisto tunnistaa henkilön ja PrimeSense-sensorin avulla pelaajasta luodaan luurankomalli, jonka mukaan ruudulla oleva avatar-hahmo muodostetaan.

Kuva 11. Hahmo liikkuu eteenpäin kärryssä, joka kulkee kaivoksen raiteella. Kuntouttaja voi valita jalokivien poimimisjärjestyksen kolmesta eri vaihtoehdosta.

Käyttäjätutkimuksessa peli sai testihenkilöiltä myönteistä palautetta ja he pitivät sitä hauskempana sekä mielekkäämpänä kuin kuntosalilla liikkumista. Kuntouttajat pitivät yleisesti hyvänä puolena sitä, että peli mahdollisti liikkumisen yksilöllisemmän muokkauksen. Vaikka pelissä törmättiin edelleen puutteellisuuksiin, lähinnä kalibrointiosuuden vaikeuteen, puoltavat tulokset kuitenkin uudenlaisten tapojen kehittämistä.

Elevuorovaikutuksen kehityksen voidaan kuvitella auttavan ylipäänsä fyysisistä rajoitteista kärsivien käyttäjien arkielämää. Eletunnistuksella parannetaan saavutettavuutta, sillä henkilö voisi käyttää kodin laitteitaan muillakin tavoin kuin käsiään käyttämällä, esimerkiksi pään asennoilla tai silmänliikkeillä ohjaten.

Edan ja muut [2009b] ovat kehittäneet kirurgeille suunnatun Gestixin (ks. kuva 12), jolla nämä voivat tarkastella magneettikuvia steriilisti leikkaushuoneessa. Gestix tehostaa kirurgien työskentelyä, sillä kosketuskäyttöliittymää

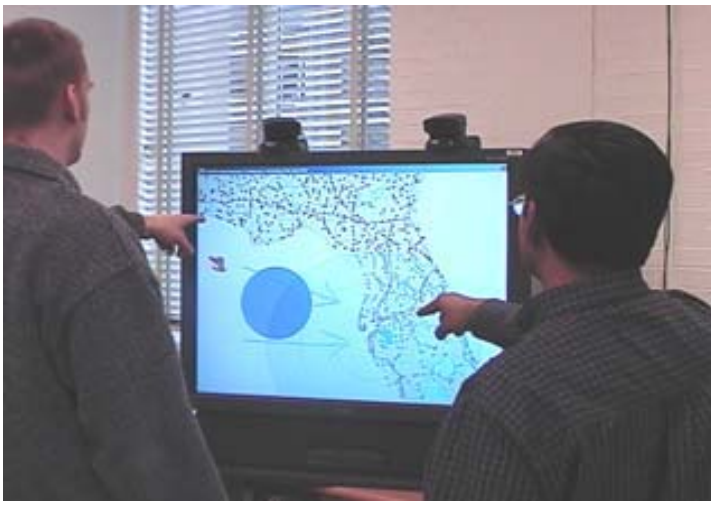
käytettäessä heidän tarvitsi laittaa joka kerta erityiset suojat käteensä ja liikkua monitorin ääreen selaamaan ja tutkimaan magneettikuvia. Käytettävyystesteistä selveni, että Gestix on tehokas, helposti käytettävä ja nopeasti opittava työkalu kirurgeille, sillä he ovat työnsä takia taitavimpia käsielein toimimisessa ja suosivat käsillä elehtimistä muita modaliteetteja enemmän.

Kuva 12. Gestix erottaa kirurgin käden muusta taustasta värisegmentoinnilla. Selatakseen kuvia lääkärin tarvitsee liikuttaa kättä nopeasti ylös, alas, oikealle tai vasemmalle. Kuvien suurennukseen ja pienennykseen kättä pyöritetään myötä- ja vastapäivään. [Edan et al., 2011]

5.5. Kriisinhallinta ja katastrofiapu

Edan ja muut [2011] nostavat esille eräänä tärkeänä elekäyttöliittymien osa-alueena kriisinhallintajärjestelmät ja katastrofin jälkityötä tehostavat järjestelmät. Laajamittaista tuhoa aiheuttavia luonnonkatastrofeja, kuten pyörremyrskyjä, maanjäristyksiä tai tulvia kohdatessa turvallisuudesta vastaavalta henkilöstöltä vaaditaan nopeaa reagointia ja täsmällistä päätöksentekoa.

DAVE_G eli Dialogue-Assisted Visual Environment for Geoinformation (ks. kuva 13) on Agrawalin ja muiden [2002] kehittämä multimodaalinen puhetta ja eleitä ymmärtävä käyttöliittymä, jonka avulla useat käyttäjät voivat samanaikaisesti käsitellä geospatiaalista dataa. Käyttäjät voivat puheen ja eleiden yhteistyönä manipuloida näytöllä olevaa karttaa. Esimerkiksi hirmumyrskyn evakuointioperaatiota suunnitellessa on mahdollista rajata vaaravyöhykkeellä olevat asuinalueet ennaltamäärättyjen avainsanojen avulla ja elein osoittamalla saada näkyville asuinalueen lähimmät pelastuslaitokset. Eleiden tunnistuksen avulla kartan käsittelyn uskotaan olevan nopeampaa kuin näppäimistöllä ja hiirellä. Usean käyttäjän tuki tekee työskentelystä sulavampaa, koska tarvittavan tiedon hakeminen ei ole yhden työntekijän käsissä vaan henkilöt voivat joustavasti manipuloida dataa samanaikaisesti.



Kuva 13. Kaksi henkilöä käyttää DAVE_G-sovellusta. [Agrawal et al., 2002]

6. Vuorovaikutuksen suunnittelu

Käsitteenä vuorovaikutuksen suunnittelu on laaja ja se pitää sisällään käytettävyysseikkojen lisäksi monia muitakin huomioitavia asioita. Käyttäjä- ja käyttökokemusta on alettu entistä enemmän painottaa, sillä miellyttävältä näyttävä ohjelma ja tyytyväinen olo sitä käytettäessä ovat merkittävä osa tuotteen laatua. Keskeisimpänä asiana on kuitenkin se, että käyttöliittymä on kanava, jonka kautta käyttäjä voi kommunikoida koneen kanssa. Jos käyttäjä ei pysty helposti oppimaan kyseistä kommunikointityyliä, on tällöin vaarana, ettei sovelluksen läheskään koko potentiaalia oteta käyttöön. Elekäyttöliittymät eivät interaktiivisuuden suunnittelussa ole erityisessä asemassa vaan niiden kehittämiseen pätevät samat periaatteet kuin vanhojenkin käyttöliittymien suunnitteluun.

Saffer [2009] ohjeistaa, että ensimmäinen kysymys, joka suunnittelijan täytyy itselleen esittää on: “Pitäisikö tämän edes olla elekäyttöliittymä?”. Elepohjainen vuorovaikutus ei ole ihanteellinen jokaiseen tilanteeseen. Hiiri ja näppäimistö ovat yhä tarkempia ja nopeampia välineitä useimpien tehtävien kohdalla. Kirjoittaminen tai numeroiden käsittely ovat esimerkkejä tilanteista kontaktittomissa käyttöliittymissä, joihin eleiden käyttö ei välttämättä ole sopiva ratkaisu. Granum ja muut [2004] esittävätkin, että suunnittelun päämääränä ei pitäisi olla jokaiselle sovellukselle toimivan, yleisen elekäyttöliittymän toteuttaminen vaan entistä tehokkaamman käyttöliittymän kehittäminen tiettyä sovellusta varten. Ei ole mielekästä hylätä vanhoja ja perinteisiä vuorovaikutustapoja, joihin ihmiset ovat jo tottuneet vain siksi, että voitaisiin tehdä vaikuttava sovellus, joka hyödyntää viimeisintä huutoa olevaa tekniikkaa. Elevuorovaikutuksen tulisi toimia muiden tapojen rinnalla ja pyrkiä parantamaan entisten vuorovaikutuskeinojen puutteellisuuksia.

Vuorovaikutuksen suunnittelun tarkastelu aloitetaan ensin esittelemällä

suunnittelumenetelmiä. Tämän jälkeen siirrytään pohtimaan elekäyttöliittymien suunnittelua käytettävyys- ja käyttökokemustekijöiden näkökulmasta.

6.1. Suunnittelumetodeja

Elekäyttöliittymien suunnitteluprosessit eivät pohjimmiltaan eroa graafisten käyttöliittymien suunnittelusta. Edelleen tarvitaan sovellusalueen taustatutkimusta, vaatimusmäärittelyä ja käyttäjätestausta. Uusi vuorovaikutustapa tuo tullessaan tosin lisää suunnitteluhaasteita, joihin ei perinteisten käyttöliittymien kohdalla ole ollut tarvetta keskittyä.

Elehtimisen fyysinen rasitus vaatii erityistä painotusta. Itse eleiden mielekkyyden lisäksi suunnittelussa on huomioitava tunnistuksen tarkkuus, joka on tärkeä osatekijä käyttökokemuksen syntymisessä ja sovelluksen mieltämisessä käyttökelpoiseksi. Elekäyttöliittymien suunnittelun osatavoitteena pitäisi olla myös pyrkimys luonnollisuuteen, mutta eleiden testaus keinotekoisissa olosuhteissa ei välttämättä tuota oikeanlaisia tuloksia, vaan käyttäjiä on havainnoitava todellisessa ympäristössä. On myös huomioitava, että suunnittelija ei välttämättä ole erilaisten tunnistustekniikoiden asiantuntija eikä tiedä, millaisia rajoituksia eleiden tunnistuksessa voidaan vielä kohdata.

6.1.1. MAGIC - Multiple Action Gesture Interface Creation

Ashbrook ja Starner [2010] ovat kehittäneet suunnittelijoille tarkoitetun MAGIC-ohjelman, jonka tavoitteena on tehostaa suunnitteluprosessia tarjoamalla mahdollisuuksia sekä asiantuntijamaiseen että noviisimaiseen työskentelyyn tunnistustekniikoiden parissa. MAGICilla pyritään myös ohjaamaan iteratiiviseen suunnitteluprosessiin ja tukemaan jatkuvaa testausta. Lisäksi ohjelmalla avitetaan suunnitelmien jälkitarkastelua muun muassa mahdollistamalla videoiden tallentaminen, jotta suunnittelija pystyy palaamaan takaisin aiemmin kehittämiensä eleiden pariin.

Ashbrook ja Starner ovat kehittäneet MAGICin osaksi kolmivaiheista suunnitteluprosessia. Ensimmäisessä vaiheessa määritellään vaatimukset ja tehdään käyttäjätutkimuksia. Toisessa vaiheessa suunnitellut eleet yhdistetään sovelluksen vaatimiin toiminnallisuuksiin ja suoritetaan alustavia käyttäjätestauksia. Viimeisessä vaiheessa tehdään lopullinen testaus ja viimeistellään tuote. MAGIC on tarkoitettu tukemaan nimenomaan prosessin toista vaihetta.

MAGICin kehittäjät ovat luoneet ohjelman käyttämiselle kolme eri työvaihetta: eleen luominen, eleen testaaminen ja väärän positiivisen ratkaisun testaaminen, jolla pyritään välttämään sellaisten eleiden valintaa, jotka saattavat aktivoida odottamatonta ja tarkoituksetonta toiminnallisuutta. Eleen luomisessa suunnittelija tekee erilaisille eleille luokat ja nauhoittaa näille esimerkkieleen tai mahdollisesti useammankin esimerkin eleen ulkoasun vaihtelun

vuoksi. Ohjelma arvioi käyttäjän puolesta kunkin luokan hyvyyden vertaamalla luokkien eleiden vastaavuuksia keskenään ja luokkien eroavuutta toisistaan sekä eleiden johdonmukaisuutta. Jokaisen luokan ja otoksen hyvyysarvot esitetään prosenttilukuina, virhematriisien avulla sekä erilaisin graafikoin. Eleen testaamisvaiheessa suunnittelija voi kokeilla, kuinka konenäkö voi tunnistaa aiemmin tehdyt eleet ja niiden luokittelut suorittamalla suunniteltuja liikkeitä kameran edessä.

Viimeisessä työvaiheessa suoritetaan väärän positiivisuuden testaus, jolla tarkoitetaan sitä, että tietty ele tuottaa toiminnan, joka ei ole odotettu eikä haluttu. Tätä vaihetta tukemaan MAGIC-ohjelma sisältää Everyday Gesture Libraryn (EGL), johon on valmiiksi tallennettu erilaisia jokapäiväisiä eleitä. EGL:n avulla suunnittelija voi verrata kehittämiään eleitä jokapäiväisiin eleisiin ja ohjelma kertoo, kuinka samanlaisia suunnitellut eleet ovat verrattuna EGL:n sisältämiin eleisiin. Tämän toiminnallisuuden ansiosta voidaan jo suunnitteluvaiheessa väistää mahdolliset odottamattomat eleiden vastaavuudet ja luoda paremmin erottuvia eleitä vastaamaan haluttua toimintoa. Ohjelman sisältämän elekirjaston avulla vähennetään myös aikaa ja tarvetta testaamiseen käyttäjien kanssa tehostaen näin koko toteutusprosessia.

6.1.2. Wizard of Oz -menetelmää hyödyntäviä suunnitteluprosesseja

Wizard of Oz -menetelmällä tarkoitetaan metodia, jossa testattava käyttöliittymä on itse asiassa vain näennäisesti toimiva. Testihenkilö suorittaa annettuja tehtäviä, mutta kaikki funktionaalisuus ja niiden seuraukset on itse asiassa "verhon takana" toimivan henkilön manuaalisesti tekemiä. Käyttöliittymä ei siis ole vielä oikeasti toimiva ja automaattinen järjestelmä.

Granum ja muut [2004] asettavat suunnitteluprosessinsa lähtökohdiksi ihmislähtöisen ja käyttäjäryhmä- sekä sovelluskohtaisen suunnittelun, sillä heidän näkökantansa mukaan ei ole olemassa universaalia "elesanastoa" *(gesture vocabulary)*. Teknisiin vaatimuksiin pohjatuvaa suunnittelua he pitävät ongelmallisena, sillä tästä lähtökohdasta käsin ei saavuteta helpoimpien, intuitiivisimpien, loogisimpien ja ergonomisimpien eleiden kehitystä.

Granumin ja muiden [2004] suunnitteluprosessi jakaantuu neljään eri vaiheeseen: funktioiden etsimiseen, käyttäjätesteihin, analyysivaiheeseen ja lopulliseen testaamiseen, josta he käyttävät termiä *benchmark.*

Ensimmäisessä vaiheessa suunnittelijan on määritettävä käyttöliittymän eri toiminnallisuudet ja ne, joiden käytössä eleillä suoritettavaa vuorovaikutusta tarvitaan. Tärkeää on löytää toimintojen vähimmäismäärä ja pyrkiä löytämään tilanteita, joissa sama ele voisi kontekstista riippuen toimia eri tavalla.

Käyttäjätestausten tavoitteena on löytää eri toiminnallisuuksia vastaavat

eleet. Tässä suunnitteluvaiheessa hyödynnetään skenaarioiden käyttöä. Eri skenaarioissa testihenkilö suorittaa aiemmin määriteltyjä toimintoja Wizard of Oz -tyyppisellä käyttöliittymällä ja koko testitilanne videoidaan. Vaihtoehtoisesti Wizard of Oz -käyttöliittymä voidaan jättää pois ja eleillä voidaan kommunikoida joko testin moderaattorin tai toisen testihenkilön kanssa.

Analyysivaiheessa poimitaan sopivimmat eleet videomateriaalin avulla. Testihenkilöiden käyttämistä eleistä arvioidaan erilaisia ominaisuuksia, joiden perusteella valinta suoritetaan. Tälläisiä ominaisuuksia ovat muun muassa poikkeavuus neutraalista asennosta ja tietyn eleen kesto sekä sen käytön frekvenssi.

Viimeisessä eli *benchmark*-vaiheessa testataan valikoitua eleiden joukkoa. Tässä vaiheessa arvoidaan semanttista tulkintaa, yleisyyttä, intuitiivisuutta, muistettavuutta, oppimisen astetta ja eleen aiheuttamaa rasitusta. Tässä prosessin osassa testihenkilöt suorittavat kolmivaiheisen kokeen ja koe pisteytetään (mitä matalampi tulos, sen parempi).

Kokeen ensimmäisessä vaiheessa testihenkilölle annetaan lista funktioista. Testattavalle esitetään ele ja tämän on arvattava, mikä listassa oleva toiminto vastaa elettä. Toisessa vaiheessa testataan muistettavuutta ja varmistetaan, että eleet ovat ymmärrettäviä. Testattavalle annetaan eleen nimi ja testihenkilön on suoritettava kyseinen ele. Viimeisessä vaiheessa arvioidaan eleiden aiheuttamaa rasitusta. Testihenkilö suorittaa eleiden sarjan useamman kerran ja antaa rasituksesta subjektiivisen arvion välillä 1 = ”Ei ongelmaa” ja 5 = ”Mahdoton”.

Akers [2007] esittelee toisenlaisen suunnittelumetodin, jonka osana Wizard of Oz -strategiaa hyödynnetään. Akersin ehdottamassa gesture-brainstorming -suunnitteluprosessissa tuotetaan niin monta ideaa kuin on vain mahdollista rajaamatta vielä vaihtoehtoja. Prosessi etenee siten, että suunnittelija toimii ”wizard”-roolissa ja loppukäyttäjä keksii eleen, jonka tämä sitten selittää suunnittelijalle. Tämän jälkeen suunnittelija simuloi funktion käyttöliittymässä. Kyseessä on iteratiivinen prosessi, jossa suunnittelija ja loppukäyttäjä käyvät läpi useamman istunnon ja aina aiempien istuntojen tuloksia hyödyntäen kehitetään uusi joukko eleitä.

Akersin suunnitteluratkaisun perusteena on huomio, että eleet sisältävät yksilöiden henkilökohtaisia ja alitajuntaisia merkityksiä. Aivoriihen tavoitteena on kehittää jokaisen käyttäjän yksilöllinen elekieli ja havainnoimalla sitä, mitä ihmiset tekevät ennemmin kuin mitä ihmiset sanovat saadaan valikoitua sopivimmat eleet lopullista käyttöliittymää varten.

6.1.3. Metodi optimaalisen elejoukon valinnalle

Edan ja muut [2009a] esittävät analyyttisen lähestymistavan optimaalisen elesanaston valinnalle. Optimoinnin päämääränä on löytää eleiden joukko, joka

on samalla sekä intuitiivinen että miellyttävä, mutta myös eleentunnistuksessa mahdollisimman tarkka. Kun muut suunnitteluohjeet painottavat, että eleiden valinta on kehitettävä ainoastaan tiettyä tarkoitusta ja sovellusta varten, Edan ja muut [2009a] päinvastoin pyrkivät yleispätevän metodin esittämiseen, jossa matemaattisten funktioiden avulla voidaan tarkkaan määrittää tarvittava elesanasto. Tarkastelematta yksityiskohtaisesti esitettyjä kaavioita käydään tässä läpi metodologian avainkohdat.

Optimaalisen eleiden joukon määrittäminen on jaettu neljään eri moduuliin. Ensimmäisessä moduulissa määritetään sovelluskohtaisesti vaaditut komennot ja jokaisen eri komennon yleisyys. Tässä vaiheessa luodaan myös suuri joukko erilaisia mahdollisia eleitä, josta Edan ja muut [2009a] käyttävät nimitystä 'Large gesture master set'. Eleiden tuottaminen tehdään yhdessä käyttäjien kanssa, jotka valitsevat mielestään sopivimman kutakin komentoa vastaavan eleen. Näin saadaan tuotettua erilaisia vaihtoehtoja, joita myöhemmässä vaiheessa aletaan karsia. Komentojen yleisyyden määrittämiseksi järjestetään testi, jossa käyttäjät suorittavat sovellukseen suunniteltuja tehtäviä. Tilannetta analysoimalla selvitetään, kuinka usein kutakin komentoa tullaan käyttämään.

Eleen intuitiivisuus määritellään sen käytön yleisyytenä ja jokaiselle elekomento -parille lasketaan niiden välisen assosiaation vahvuus. Määrittämällä ennalta kynnysarvo intuitiivisuudelle, voidaan 'Large gesture master set':ista poistaa kynnysarvoa pienempiä assosiaatiovahvuuksia saadut eleet ja näin kaventaa mahdollisten eleiden määrää. Tästä uudesta eleiden joukosta käytetään nimitystä 'The gesture master set'. Koko elejoukon aiheuttama rasitus ja miellyttävyys lasketaan määrittämällä eri eleiden välillä liikkumisen fyysinen vaikeus, kuinka kauan eleestä toiseen siirtyminen kestää ja kuinka usein komennosta toiseen siirrytään.

Kun ensimmäisessä moduulissa määritettiin käyttäjäkeskeisiä tekijöitä, toisessa moduulissa keskitytään koneesta riippuvaan faktoriin. Tässä vaiheessa etsitään iteratiivisesti eleiden alajoukkoja, jotka saavuttavat ennalta määritellyn ja halutun tarkkuustason tunnistamisessa. Prosessissa käytetään tunnistusalgoritmia, joka etsii erilaisia joukkoja aiemmassa vaiheessa muodostetusta optimoidusta pääjoukosta *(master set).*

Kolmannessa moduulissa etsitään jokaiselle komennolle yksi sitä vastaava ele. Tavoitteena on yhdistää eri komennot ja löydetyt eleiden alajoukot siten, että käyttäjäkeskeiset tekijät (intuitiivisuus, miellyttävyys) saavat maksimiarvot. Näistä muodostetaan erilaisia elejoukkojen ratkaisuvaihtoehtoja. Viimeisessä moduulissa etsitään muodostetuista elejoukkojen vaihtoehdoista Pareto-optimaalinen ratkaisu, joka määrittää elejoukon, jossa minkään tekijän ominaisuutta ei voida enää parantaa huonontamatta jotain toista tekijää. Tällöin on

päädytty optimaalisimpaan ratkaisuun.

Metodin eräänä heikkoutena tosin on se, että nyt lopullisessa ratkaisussa yksi ele vastaa aina yhtä komentoa. Joustavamman ratkaisun aikaansaamiseksi voitaisiin huomioida se seikka, että yksi ele voi vastata useampaa eri komentoa riippuen siitä, missä tilanteessa sitä käyttöliittymässä käytetään. Elekomentojen päällekkäisyys otetaan huomioon muun muassa aiemmin esitellyssä Granumin ja muiden [2004] Wizard of Oz -menetelmässä.

6.2. Käytettävyys ja käyttökokemus

Jakob Nielsenin [1993] määritelmän mukaan käytettävyys koostuu pohjimmiltaan viidestä eri osatekijästä: opittavuudesta, tehokkuudesta, muistettavuudesta, mielekkyydestä ja mahdollisimman vähästä määrästä virheitä käyttäjälle. Näiden laatuominaisuuksien lisäksi käytettävyyteen vaikuttavat hyödyllisyys *(usefulness)* ja käyttökelpoisuus *(utility)*. Lisäksi sosiaalisen ja käytännöllisen hyväksyttävyyden ulottuvuudet ovat osa käytettävyyttä. Käytettävyyskriteerien lisäksi vuorovaikutusta suunniteltaessa on otettava huomioon käyttäjien tarpeiden tunnistaminen ja käsitemallien tukeminen. Onnistuneen vuorovaikutuksen syntymiseen vaikuttavat myös käyttö- ja käyttäjäkokemus.

Elekäyttöliittymiä suunnitellessa tavoitteet eivät ole perusteiltaan erilaisia verrattuna graafisten käyttöliittymien suunnitteluun. Näkökulman muutos tosin haastaa suunnittelijat löytämään ratkaisuja ongelmiin, joita ei aikaisemmin ole osattu ajatella ja löydetyt ratkaisut saattavat muuttaa totuttuja vuorovaikutusmalleja merkittävästikin. Seuraavassa vuorovaikutuksen suunnittelua lähestytään pohtimalla esitettyjen kriteerien täyttymistä elekäyttöliittymien kohdalla.

6.2.1. Opittavuus

Elekäyttöliittymiä vaivaa niiden näkymättömyys, joka tarkoittaa sitä, että käyttäjälle ei tarjota visuaalisia tai auditiivisia vihjeitä siitä, mitä tämän tekemästä eleestä seuraa ja mikä ylipäänsä on oikea ele missäkin tilanteessa [Bernardes et al., 2009; Nielsen and Norman, 2010]. Tässä mielessä perinteiset graafiset käyttöliittymät huomioivat käyttäjänsä paremmin. Graafisen käyttöliittymän vahvuutena on opiskeltavuus. Ikoneita, painikkeita ja valikoita tutkimalla käyttäjä pystyy selvittämään, mitä tämän on mahdollista tehdä ja miten hän voi saavuttaa haluamansa.

Elekäyttöliittymissä henkilön on kuitenkin usein tiedettävä ennalta, minkälaisia eleitä kone pystyy tunnistamaan. Nielsen ja Norman [2010] kritisoivatkin, ettei oikeanlaisen oppimisen pitäisi tapahtua siten, että käyttäjä sattumanvaraisesti ja yllätyksellisesti löytää oikean komennon tai siten, että toinen käyttäjä kertoo, että tietty ele toimiikin tietyssä kontekstissa tietyllä tavalla.

On itsestään selvää, että johdonmukaisuus minkä tahansa käyttöliittymän suunnittelussa on tärkeää, mutta tällä hetkellä ollaan tilanteessa, jossa varsinkin yritysmaailmassa kilpaillaan siitä, kuka keksii uuden vuorovaikutustavan säännöt ja kenen luomuksia aletaan jatkokehittää. On hyvä, että uusilla menetelmillä leikitään ja niitä koetellaan, jotta löydettäisiin lopulta paras vaihtoehto, mutta Norman ja Nielsen [2010] painottavat sitä, että kehitystyö kuuluu käytettävyyslaboratorioihin eikä kuluttajamarkkinoille. Alusta pitäen sovellusten toteuttamiseen kaivattaisiin yhteneväisiä suunnitteluohjeita. Tukeaksemme käyttäjän käsitemallien muodostumista olisi toivottavaa standardoida elekomentoja ainakin toiminnoille, jotka ovat yhteisiä kaikille sovelluksille, ettei käyttäjän tarvitsisi jokaisella kerralla selvittää, kuinka kyseinen käyttöliittymä toimii.

6.2.2. Muistettavuus

Eleiden määrää valittaessa on otettava huomioon ihmisen muistin rajallisuus ja alttius virheille. Eleiden määrälle ei voida määrittää yhtä kaikissa tilanteissa paikkansa pitävää sääntöä vaan eleiden valintaan vaikuttavat useat erilaiset tekijät. On huomioitava muun muassa yksilölliset erot kognitiivisissa kyvyissä, käytettävien eleiden intuitiivisuus ja minkälaiselle sovellusalueelle käyttöliittymää ollaan suunnittelemassa.

Bernardes ja muut [2009] luottavat Millerin maagiseen lukuun 7 ± 2 ja ehdottavat, että noin 5-10 elettä olisi sopiva määrä käyttöliittymän eri konteksteihin, mutta ei välttämättä koko sovelluksen komentojen määräksi. Muistikyvyn rajallisuutta voidaan toisaalta venyttää valitsemalla helposti omaksuttavia ja jo käytössä olevia eleitä. Lisäksi eleitä voidaan käyttää samoin kuin esimerkiksi hiiren oikeaa näppäintä. Kontekstista riippuen ele tuottaa erilaisen tuloksen.

Käyttäjille tarjottava mahdollisuus muokata ja luoda eleitä itselle sopiviksi voisi olla eräs keino, jolla mentaalista kuormitusta voitaisiin vähentää. Wexelblat [1998] näkee tässä kaksi myönteistä etua. Ensinnäkin muokattavilla eleillä otetaan huomioon kulttuuriset erot ja toiseksi eleiden tunnistustarkkuutta voidaan parantaa. Edan ja muut [2011] huomauttavat tosin, ettei käyttäjälle kannata antaa loputtomasti muokkausvaihtoehtoja vaan suunnittelijan on harkittava, mitkä komennot on tärkeää jättää ennaltamäärätyiksi ja mitä komentoja käyttäjän on mahdollista muuttaa, jotta käytön joustavuus olisi kuitenkin riittävää. Teknisesti tällainen toiminnallisuus ei ole mahdotonta toteuttaa. Muun muassa piilotetun Markovin mallin avulla konetta voidaan hyvin opettaa tunnistamaan erilaisia eleitä varsinaisen käytön aikana, ja kone parantaa eleen tunnistustarkkuutta jokaisella käyttökerralla. Alussa tunnistus ei välttämättä ole erityisen tarkkaa, mutta mitä enemmän elettä käytetään, sitä

tarkemmaksi tunnistus muuttuu.

6.2.3. Staattiset vai dynaamiset eleet?

Ihmisen elehdinnälle on luonnollista jatkuva liike. Toteutuksen näkökulmasta staattisten eleiden tunnistaminen voi olla helpompaa ja tarkempaa, mutta ele ei tällöin välttämättä ole kovin intuitiivinen. Vuorovaikutus voitaisiin toteuttaa myös siten, että olisi jokin ele, jolla ilmaistaan komennon alkaminen ja lopettaminen. Näiden kahden eleen välissä käyttäjä tekisi dynaamisen komennon.

Wexelblat [1998] kyseenalaistaa kokonaan tämäntapaisen suunnittelun. Hänen mielestään tarkempi ja yksinkertaisempi tapa olisi tukeutua näissä tilanteissa edelleen perinteiseen laitteistoon, sillä jos estetään luonnollinen tapa käyttää eleitä, ei ole mitään syytä vaikeuttaa toimintaa vaatimalla monimutkaisia eleiden sarjoja.

Grandhi ja muut [2011] ovat myös dynaamisten eleiden kannalla ja puhuvat pantomiimisten sekä havainnollisten eleiden hyödyntämisestä. Ensiksikin manipulatiivisille toiminnoille täytyisi suunnitella eleitä, jotka ikonisesti esittävät kyseistä toimintoa. Esimerkiksi kohteen poistaminen voitaisiin esittää pyyhkäisyeleellä eikä staattisella käsien asennolla, esimerkiksi kädet ristissä. Ihmiset eivät myöskään pidä intuitiivisena käyttää kehon osaa kuvaamaan eri esineiden käyttöä. Luonnollisempaa on esittää esimerkiksi vasaroimista pantomiimisesti aivan kuin työkalu todella olisi käyttäjän kädessä. Lisäksi, jos käyttöliittymässä on manipuloitava jossakin tilassa sijaitsevia objekteja, olisi mielekästä tukea kahden käden elehtimistä. Grandhin ja muiden [2011] mukaan kahdella kädellä saadaan parempi käsitys virtuaalisesta tilasta käyttämällä toista kättä apuna tilasuhteiden hahmottamisessa ja toista kättä varsinaisen toiminnon välittäjänä.

Toisaalta staattisten eleiden käyttäminen voi olla mielekästä, jos eleelle on vakiintunut merkitys jokapäiväisissä tilanteissa. Esimerkiksi pysäyttämistä tarkoittava ele, jossa käsi suoristetaan eteenpäin näyttäen kämmentä, on monille käyttäjäryhmille tavanomainen ele. Jos käyttöliittymän funktiolle ei ole kuitenkaan olemassa sitä vastaavaa käden asentoa, tulisi komennolle kehittää dynaaminen ja täten toimintoa kuvaamampi ele.

6.2.4. Virheet, tarkkuus ja tehokkuus

Kun käyttäjällä on muistettavanaan eri toiminnoille useita erilaisia eleitä, alttius virheille lisääntyy. Koska eleet eivät jätä jälkiä, käyttäjän on vaikea ymmärtää virheen syytä, mikä johtaa epätoivottuun kontrollin menetyksen tunteeseen [Nielsen and Norman, 2010]. Jotta käyttäjän virheiden tekeminen voitaisiin estää, Norman [2010] suosittelee tukeutumista perinteisten graafisten käyttöliittymäelementtien käyttämiseen, kuten kumoamisoperaation tarjoamiseen ja

opasteiden hyödyntämiseen. Yksi tapa estää väärän eleen tekemisen mahdollisuus on tarjota visuaalisia vihjeitä eleistä, jotka kyseisessä kontekstissa ovat oikeita. Vihjeet voisivat olla kuvia tai animaatioita, jotka olisivat myös ikään kuin muistilappuja käyttäjälle.

Vihjeiden avulla käyttäjää voidaan ohjata oikeiden komentojen valintaan, mutta teknisiä virheitä ei tälläkään tavalla voida estää. Mikäli kone ei kykene ollenkaan tunnistamaan virhettä, tällöin on suotavaa ilmoittaa siitä käyttäjälle joko visuaalisesti tai lisäksi äänipalautteella. Dialogi-ikkunoiden käyttö ei välttämättä ole käypä ratkaisu palautteeksi, vaan sopivampia voisivat olla yleistajuisten symbolien ja värien käyttö sekä virheen ilmaisevien äänien käyttö. Mikäli kone tulkitsee käyttäjän tarkoituksellisen eleen väärin ja suorittaa odottamattoman toiminnon, pitäisi käyttäjällä olla mahdollisuus ”undo”-eleen suorittamiseen. Kumoamisoperaatio on eräs niistä toiminnoista, jolle alusta pitäen kaivattaisiin vakiintunutta elettä.

Näiden tunnistusvirheiden lisäksi elepohjaisessa vuorovaikutuksessa törmätään ongelmaan, jota Edan ja muut [2011] kutsuvat immersiosyndroomaksi. Sillä viitataan ongelmaan, jossa kone voi tulkita käyttäjän tahattomatkin eleet komennoiksi. Käyttäjä voi myös samaan aikaan elehtiä toisille henkilöille, jolloin kone voi tulkita väärin käyttäjän liikkeet. Myös taustalla liikkuvien ihmisten liikkeet voivat häiritä eleiden tunnistusta. Tällaisia tilanteita varten voisi olla sopivaa tarjota mahdollisuus eletunnistuksen pysäyttämiselle, jotta käyttäjän muu toiminta ei häiritsisi sulavaa toimintaa koneen kanssa.

Bernardes ja muut [2009] huomauttavat, että käyttöliittymän asentamiseen kuluvan ajan pitäisi olla mahdollisimman lyhyt, sillä ihmiset haluavat nopeasti aloittaa varsinaisen työskentelynsä. Käyttöliittymän ei siis pitäisi vaatia erityisiä vaatimuksia valaistukselle tai työskentelytilan taustalle, eikä ennen jokaista käyttöönottoa pitäisi olla tarpeellista suorittaa monimutkaista kalibrointia.

Elevuorovaikutus on lähes aina hitaampaa kuin näppäimistöllä ja hiirellä työskentely [Bernardes et al., 2009]. Jo 300 millisekunnin responssiviiveen kohdalla järjestelmä alkaa vaikuttamaan hitaalta ja jähmeältä [Edan et al., 2011]. Itse eleen suorittamiseen ja sen tunnistamiseen voi kulua merkittävästikin enemmän aikaa, joten elevuorovaikutus ei vielä sovellu esimerkiksi nopeita refleksejä vaativiin peleihin [Bernardes et al., 2009].

6.2.5. Fyysinen kuormitus

Elehtimisen viehätyksen kääntöpuolena on sen vaatima fyysisyys. Käsien heiluttaminen tai jaloilla potkiminen voi lyhyenkin käytön jälkeen rasittaa myös perusterveitä ja nuoria ihmisiä. Cabral ja muut [2005] tutkivat

elekäyttöliittymien käytettävyyttä virtuaaliympäristöissä. Tutkimukseen osallistuneista henkilöistä, joiden keski-ikä oli 25 vuotta, 40% valitti käsien väsymistä käytön aikana. Jos henkilö on vanha tai jollain tavalla fyysisesti rajoittunut, fyysinen kuormitus voi entistä enemmän vaikeuttaa käyttöliittymän käyttöä.

Yksilöllisten piirteiden lisäksi lihasrasitukseen voivat vaikuttaa myös ulkoiset tekijät, kuten lämpötila tai elehtijän asento, jotka vaikeuttavat ennakoitavuutta suunnittelussa [Edan et al., 2011]. Sekä staattiset asennot että dynaamiset eleet rasittavat kehoa. Vakaat eleet rasittavat asennon ylläpitämisen takia ja dynaamiset eleet liikkeen vaatiman fyysisyyden vuoksi. Käytetään sitten kumpaa tahansa tyyliä, molempiin pätee yksi periaate: liiallista toistoa on vältettävä. Sen lisäksi tulisi välttää ylisuurten eleiden suunnittelua, jotta ele sopisi useammille käyttäjäryhmille ja fyysinen kuormitus olisi mahdollisimman pientä.

6.2.6. Kulttuuriset vaikutukset ja sosiaalinen hyväksyttävyys

Elekäyttöliittymien suunnittelussa väistämättä kohdataan kulttuurierojen vaikutukset sovelluksen toteuttamiseen. Samoille eleille voidaan antaa eri kulttuureissa hyvinkin päinvastaisia merkityksiä. Esimerkiksi “peukut ylös” -eleelle annetaan länsimaissa positiivinen merkitys, mutta aasialaisissa ja islamilaisissa kulttuureissa ele tulkitaan loukkaavaksi ja töykeäksi. Samoin “ookoo”-ele, jossa etusormi ja peukalo pidetään yhdessä ja muut sormet suoristetaan osoittamaan ylöspäin, saa toisissa kulttuureissa merkityksen, että kaikki on hyvin tai jokin asia on erinomaista, mutta joissakin Etelä-Euroopan ja Etelä-Amerikan maissa kyseinen ele tulkitaan hävyttömäksi ja pahennusta herättäväksi.

Kulttuuriset merkityserot asettavat suunnittelijoille haasteita eleiden valinnan suhteen. Granum ja muut [2004] ehdottavat, että samoin kuin perinteisissä ohjelmissa on olemassa kielipaketteja, voitaisiin elekäyttöliittymiin muokata erilaisia elejoukkoja, jotka vastaisivat tietyn maan tai kielen sääntöjä.

Kulttuuristen erojen rinnalla suunnittelussa on arvioitava myös eleiden sosiaalista hyväksyttävyyttä. Brewster ja Rico [2010] tutkivat mobiilikäyttöliittymissä käytettävien eleiden sosiaalista hyväksyttävyyttä. Tutkimuksessa käytettiin joukkoa erilaisia eleitä, sekä kosketuseleitä että kontaktittomia eleitä. Koehenkilöitä pyydettiin ensin arvioimaan, missä tilanteissa ja paikoissa (esimerkiksi kotona tai julkisella kadulla) sekä kenen seurassa (esimerkiksi ystävien tai vieraiden) he olisivat halukkaita käyttämään tietynlaista elettä. Kyselyn pohjalta valittiin kaksi hyväksyttävää ja kaksi ei-hyväksyttävää elettä ja koehenkilöiden täytyi käyttää eleitä sekä yksityisessä että julkisessa tilanteessa, huoneessa yliopistolla ja vilkkaalla kadulla.

Tuloksista ilmeni, että ele arvioitiin ja koettiin hyväksyttävämmäksi, mikäli se pystyttiin naamioimaan arkipäiväiseksi eleeksi. Lisäksi pienet eleet, joilla ei kiinnitetty huomiota itseen ja itselle mukavantuntuiset eleet koettiin hyväksyttävämmiksi. Laitteen esilläolo vaikutti myös hyväksyttävyyteen, sillä elehtimisen ei koettu vaikuttavan sivustakatsojille oudolta, koska mobiililaitteen näkyminen antoi epätavallisellekin eleelle selkeän merkityksen.

Ympäristön ja yleisön vaikutukset tietyn eleen hyväksyttävyyteen olivat melko suoraviivaisia ja odotettujakin. Mitä yksityisemmässä paikassa ja mitä läheisempiä ympärillä olevat ihmiset olivat, sitä halukkaampia koehenkilöt olivat käyttämään epätavanomaisia ja outoja eleitä. Ihmiset eivät halua nolata itseään tai näyttää hölmöltä ja varovat käyttämästä eleitä, joilla heidän toimintaansa voitaisiin tulkita negatiivissävytteisesti.

Koska puhtaasti kontaktittomat elekäyttöliittymät eivät ole vielä laajasti yleistyneet, ihmiset eivät luultavasti vielä uskalla toimia tavalla, joka ei sovi sosiaalisiin normeihin. Suunnittelussa täytyisi siis kyetä hillitsemään erikoisten eleiden valintaa ja tarjota käyttäjälle joustavuutta siten, ettei elettä ole pakko käyttää, mikäli se voidaan kokea epämiellyttäväksi. Mutta normaalin käyttäytymisen rajat ovat häilyviä ja niitä voidaan venyttää, sillä sosiaalisesti hyväksymättömästäkin toiminnasta voi tulla ajan mittaan hyväksyttävää, aivan kuten on käynyt Bluetooth-kuulokkeiden kanssa [Brewster and Rico, 2010].

7. Pohdintaa

John Underkoffler – mies, joka suunnitteli Minority Report -elokuvan ennennäkemättömän käyttöliittymän ja myöhemmin toteutti sen todellisen version G-Speakin – on sanonut, että hän ja hänen työryhmänsä eivät lopeta ennen kuin elekäyttöliittymät ovat läsnä kaikkialla, ennen kuin vuorovaikutus kaikkien tietokoneiden kanssa on muuttunut elepohjaiseksi [Underkoffler, 2010]. Uudet työskentelytavat ovat enemmän kuin tervetulleita, mutta toivottavaa on, ettei vanhojakaan tapoja täysin syrjäytetä.

Laitteiston kalleuteen elekäyttöliittymien menestys ei tule tyrehtymään. Microsoft Kinect on ollut jo huikea menestys ja on myynyt useita miljoonia kappaleita. Aiemmin esitellyn SixthSensen rakentamiseen vaadittava laitteisto ei maksa enempää kuin 350 dollaria ja tavoitteena on myös avoimen lähdekoodin projekti, jossa jokainen voi rakentaa oman SixthSense-laitteensa ja kehittää sille lisää sovelluksia. Hintojen halpuus on pitkälti kameroiden ja konenäkömenetelmien kehityksen ansiota.

Hintojen sijaan elekäyttöliittymien hyödyllisyys ja käyttökelpoisuus riippuvat pitkälti vuorovaikutuksen suunnittelusta. Kun uudenlainen vuorovaikutustapa kohdataan, on vanhat ongelmat ratkaistava uudestaan ja lisäksi saatetaan

kohdata yllättäviäkin haasteita, joihin täytyy etsiä ratkaisu. Käytettävyyden kardinaalihyveiden – opittavuuden, muistettavuuden, tehokkuuden ja mielekkyyden sekä virheettömyyden – täyttämisessä kohdataan tosin haasteita.

Käytettävien eleiden tulisi olla intuitiivisia ja helppoja muistaa, mutta on otettava huomioon se seikka, että ihmiset liittävät eleisiin omia merkityksiään ja tulkintojaan. Tähän liittyen kulttuuriset vaikutukset saavat myös entistä suuremman painotuksen.

Vaikeasti opittavan vuorovaikutusmallin suunnittelun rinnalla on varottava myös käyttäjän muistin kuormitusta vaatimalla tältä liian monen eleen oppimista. Suunnittelijalle jää harkittavaksi, riittävätkö visuaaliset ja auditiiviset palautteet tai vihjeet keventämään mentaalista taakkaa vai olisiko parempi antaa käyttäjän muokata eleitä itsensä näköiseksi. Eleiden määrän ja opittavuuden lisäksi liikkeiden rasitukseen on kiinnitettävä huomiota. Elehtiminen voi olla fyysisesti melko raskasta jo hyvin lyhyen käytön jälkeen.

Sen sijaan, että suunniteltaisiin puhtaasti eleiden tunnistukseen nojaavia käyttöliittymiä, parempi ratkaisu saattaisi olla keskittyä multimodaalisten sovellusten toteuttamiseen. Ihmisille kuitenkin on luonnollista yhdistää puhetta sekä elehtimistä ja useimmiten nämä toimivat toisiaan täydentäen. Lisäksi kosketuksen kautta olisi mahdollista hyödyntää tuntopalautetta, joka tukisi visuaalista ja auditiivista palautetta varsinkin silloin, kun muut aistikanavat häiriintyvät, esimerkiksi melussa tai heikkonäköisten käyttäjien kohdalla. On lisäksi mielenkiintoista, millä tavalla eleiden emotionaalisuutta tullaan hyödyntämään.

Paljon toivottu luonnollisuus vaikuttaa vielä kyseenalaiselta, jos luonnollisuudella tosiaan pyritään keskustelevaan vastavuoroisuuteen koneen kanssa. Luonnollisuutta nakertaa tarve opetella komennot ennalta, mikä tarkoittaa, että käyttäjän on oltava koko ajan tietoinen liikkeistään ja varottava väärien eleiden tekemistä. Ihmiselle luonnollista olisi vapaamuotoinen elehtiminen, mutta menetelmät eivät vielä ole tarpeeksi tarkkoja tulkitakseen jokaisen henkilön yksilöllisiä liikkeitä ja reagoimaan niihin tilanteen mukaisesti. Seikkaeroista huolimatta elekäyttöliittymät epäilemättä ovat hyödyllisiä monilla sovellusalueilla, ne parantavat vuorovaikutuskeinojemme arsenaalia ja ennen kaikkea muuttavat tylsän hiirellä ja näppäimistöllä työskentelyn paljon antoisammaksi kokemukseksi.

8. Yhteenveto

Tässä tutkielmassa käsiteltiin kontaktittomien elekäyttöliittymien vuorovaikutuksen suunnittelua ja eletunnistuksessa käytettyjä menetelmiä. Aluksi tarkasteltiin eleiden luokittelua tietokoneen ja ihmisen välisen vuorovaikutuksen kontekstissa. Taksonomia perustui pääosin Karamin ja Schraefelin [2005] esit-

tämään jaotteluun. Samoin elekäyttöliittymien jaottelussa tukeuduttiin Karamin ja Schraefelin [2005] näkemykseen.

Eleiden tilastollisina tunnistusmenetelminä esiteltiin piilotettu Markovin malli ja äärellinen automaatti. Konenäkömenetelmien yhteydessä käsiteltiin piirreirrotusta värin ja muodon perusteella sekä esiteltiin Microsoft Kinectin toimintaperiaate korkealla tasolla.

Vuorovaikutuksen suunnittelua lähestyttiin suunnittelumenetelmien esittelyllä ja arvioimalla käytettävyyden ominaisuuksien toteutumista elekäyttöliittymissä. Suunnittelumenetelmien yhteydessä esiteltiin MAGIC-ohjelma, kaksi Wizard of Oz -menetelmää ja viimeisenä aiemmista poikkeava metodi elesanaston valinnalle. Käytettävyyden ja käyttökokemuksen yhteydessä kartoitettiin eleiden opittavuuteen ja muistettavuuteen vaikuttavia tekijöitä. Myös virhetilanteiden ja fyysisen kuormituksen huomioonottamista käsiteltiin. Lopuksi käytiin läpi kulttuurin ja sosiaalisen hyväksyttävyyden vaikutuksia vuorovaikutuksen suunnitteluun.

Viiteluettelo

[Acharya and Mitra, 2007] Tinku Acharya and Sushmita Mitra, Gesture recognition: a survey. *IEEE T. Syst. Man Cyb.— Part C: Applications and Reviews* **37**, 3 (May 2007), 311–324.

[Agrawal et al., 2002] Pyush Agrawal, Isaac Brewer, Guoray Cai, Sven Fuhrmann, Alan MacEachren, Ingmar Rauschert, Rajeev Sharma, and Hongmei Wang, Designing a human-centered, multimodal GIS interface to support emergency management. In: *Proc. of the 10th ACM International Symposium on Advances in Geographic Information Systems (GIS '02),* 119–124.

[Akers, 2007] David L. Akers, Observation-based design methods for gestural user interfaces. In: *Proc. Of Extended Abstracts on Human Factors in Computing Systems (CHI EA '07), CHI '07, Doctoral Consortium,* 1625–1628.

[Ansari et al., 2002] Rashid Ansari, Rober Bryll, Susan Duncan, Cemil Kirbas, Xin-Feng Ma, Karl E. McCullough, David McNeill, and Francis Quek, Multimodal human discourse: gesture and speech. *ACM T. Comput.-Hum. Int.* **9,** 3 (Sept. 2002), 171–193.

[Ashbrook and Starner, 2010] Daniel Ashbrook and Thad Starner, MAGIC: A motion gesture design tool. In: *Proc. of the 28th International Conference on Human Factors in Computing Systems,* 2159–2168.

[Baker and Matthews, 2004] Simon Baker and Iain Matthews, Active appearance models revisited. *Int. J. Comput. Vis.* **60**, 135–164, 2004.

[Baker et al., 2004a] Simon Baker, Iain Matthews, and Jeff Schneider, Automatic

construction of active appearance models as an image coding problem. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**, 10, 1380–1384, 2004 (Oct.).

[Baker et al., 2004b] Simon Baker, Takeo Kanade, Iain Matthews, and Jing Xiao, Real-time combined 2D+3D active appearance models. In: *Proc. of the 2004 IEEE Computer Society, Conference on Computer Vision and Pattern Recognition*, 538–542.

[Bernardes et al., 2009] João L. Bernardes, Ricardo Nakamura, and Romero Tori, Design and implementation of a flexible hand gesture command interface for games based on computer vision. In: *Proc. of the 2009 VIII Brazilian Symposium on Games and Digital Entertainment (SBGAMES '09),* 89–98.

[Billinghurst, 2011] Mark Billinghurst, Gesture driven input. In: William Buxton, Mark Billinghurst, Yves Guiard, Abigail Sellen, and Shumin Zhai (eds.), *Human Input to Computer Systems: Theories, Techniques and Technology.*

[Blake et al., 2011] Andrew Blake, Mat Cook, Mark Finocchio, Andrew Fitzgibbon, Alex Kipman, Richard Moore, Toby Sharp, and Jamie Shotton, Real-time human pose recognition in parts from single depth images. Microsoft Research Cambridge & Xbox Incubation, June 2011.

[Bolas et al., 2011] Mark Bolas, Chien-Yen Chang, Belinda Lange, Brad Newman, Thai Phan, Evan A. Suma, and Albert Rizzo, Leveraging unencumbered full body control of animated virtual characters for game-based rehabilitation. *Lecture Notes in Computer Science* **6774**, 2011, 243–252.

[Brewster and Rico, 2010] Stephen Brewster and Julie Rico, Usable gestures for mobile interfaces: evaluating social acceptability. In: *Proc. of the 28th International Conference on Human Factors in Computing Systems (CHI '10),* 887–896.

[Cabral et al., 2005] Marcio C. Cabral, Carlos H. Morimoto, and Marcelo K. Zuffo, On the usability of gesture interfaces in virtual reality environments. In: *Proc. of the 2005 Latin American Conference on Human-computer Interaction (CLIHC '05),* 100–108.

[Chan et al., 2011] Jacky C.P. Chan, Howard Leung, and Jeff K.T. Tang, Interactive dancing game with real-time recognition of continuous dance moves from 3D human motion capture. In: *Proc. of the 5th International Conference on Ubiquitous Information Management and Communication (ICUIMC' 11),* 2011 (Feb.).

[Cootes and Taylor, 1992] T.F. Cootes and C.J. Taylor, Active shape models: 'smart snakes.' In: *Proc. of the British Machine-Vision Conference,* 1992 (Sept. 22–24), 266–275.

[Edan et al., 2009a] Yael Edan, Helman Stern, and Juan Wachs, A method for

selection of optimal hand gesture vocabularies. *Lecture Notes in Computer Science* **5085**, 57–68.

[Edan et al., 2009b] Yael Edan, Craig Feied, Michael Gillam, Jon Handler, Helman Stern, and Juan Pablo Wachs, A novel hand gesture-based image browsing system for the operating room. *Hospital Information & Technology Europe Spring* **2,** 1, (March 2009), 43–45.

[Edan et al., 2011] Yael Edan, Mathias Kölsch, Helman Stern, and Juan Pablo Wachs, Vision-based hand gesture applications. *Commun. ACM* **54**, 2 (Feb. 2011), 60–71.

[Grandhi et al., 2011] Sukeshini A. Grandhi, Gina Joue, and Irene Mittelberg, Understanding naturalness and intuitiveness in gesture production: insights for touchless gestural interfaces. In: *Proc. of the 2011 Annual Conference on Human Factors in Computing Systems (CHI '11)*, 821–824.

[Granum et al., 2004] Erik Granum, Thomas B. Moeslund, Michael Nielsen, and Moritz Störring, A procedure for developing intuitive and ergonomic gesture interfaces for HCI. *Lecture Notes in Computer Science* **2915**, 105–106.

[Haywood et al., 2011] K. Haywood, L. Johnson, A. Levine, R. Smith, and H. Willis, *The 2011 Horizon Report.* Austin, Texas: The New Media Consortium, 2011.

[Halverson et al., 2008] Tim Halverson, Anthony Hornof, Jeffrey Stolet, and Troy Rogers, Bringing to life the musical properties of the eyes. University of Oregon, Department of CIS, Technical Report **08-05**.

[Ishii et al., 1992] Kenichiro Ishii, Jun Ohya, and Junji Yamato, Recognizing human actions in time-sequental images using hidden Markov model. In: *Proc. Of IEEE Computer Society, Conference on Computer Vision and Pattern Recognition (CVPR '92)*, 379–385.

[Islam et al., 2011] Md. Zahidul Islam, Chil-Woo Lee, and Chi-Min Oh, Pictorial structures-based upper body tracking and gesture recognition. In: *2011 17th Korea-Japan Joint Workshop on Frontiers of Computer Vision*, 2011 (Feb), 1–6.

[Karam and Schraefel, 2005] Maria Karam and M.C. Schraefel, A taxonomy of gestures in human computer interaction. University of Southampton, Electronics and Computer Science, Technical report **ECSTR-IAM05-009**.

[KinectEDucation, 2011] KinectEDucation - Connected Education, http://www.kinecteducation.com/. Checked 11.12.2011.

[LiveScience, 2009] Parkinson's patients go to Wii-hab. June 11, 2009. http://www.livescience.com/3677-parkinson-patients-wii-hab.html. Checked 27.11.2011.

[McNeill, 1992] David McNeill, *Hand and Mind: What Gestures Reveal About*

Tought. Chicago: The University of Chicago Press, 1992.

[Music Hack Day, 2011] Music Hack Day Boston, 5-6 November, 2011, http://wiki.musichackday.org/index.php?title=Kinect_BeatWheel. Checked 11.12.2011.

[Myers, 1998] Brad A. Myers, A brief history of human-computer interaction technology. *Interaction*s **5,** 2 (March/April 1998), 44–54.

[Nielsen, 1993] Jakob Nielsen, *Usability Engineering.* Academic Press, 1993.

[Nielsen and Norman, 2010] Jakob Nielsen and Donald A. Norman, Gestural interfaces: a step backward in usability. *Interactions* **17,** 5 (Sept. - Oct. 2010), 46–49.

[Norman, 2010] Donald A. Norman, Natural user interfaces are not natural. *Interactions* **17,** 3 (May – June 2010) , 6–10.

[Saffer, 2009] Dan Saffer, *Designing Gestural Interfaces.* O'Reilly Media, Inc., 2009.

[Underkoffler, 2010] TED Speakers: John Underkoffler, www.ted.com/speakers/john_underkoffler.html. Checked 12.12.2011.

[Mistry, 2010] Pranav Mistry, SixthSense -integrating infromation with the real world, http://www.pranavmistry.com/projects/sixthsense/. Checked 27.11.2011.

[Wexelblat, 1998] Alan Wexelblat, Research challenges in gesture: open issues and unsolved problems. *Lecture Notes in Computer Science* **1371**, 1998, 1–11.

Avoimen lähdekoodin ominaisuudet ja potentiaali

Juha Kaura

Tiivistelmä.
Tässä tutkielmassa tarkastelen kriittisesti avoimen lähdekoodin (OS) etuja ja haittoja. Lopuksi hahmotan avoimen lähdekoodin vaikutusta tietotekniikan maailmaan ja avoimen lähdekoodin potentiaalia muuttaa nykyisiä tietotekniikan perinteitä. Tutkielman painopiste on enemmän avoimen OS-tilannekartoituksessa kuin OS:n ongelmien ratkaisemisessa.

Avainsanat ja -sanonnat: Avoin lähdekoodi, suljettu lähdekoodi, Linux, turvallisuus, modulaarisuus, pirstaloituminen, perinnejärjestelmät
CR-luokat: D.2.9, D.4

1. Johdanto

Avoimen lähdekoodin ohjelmisto (OSS) on viimeisten kahden vuosikymmenen aikana levinnyt laajempaan käyttöön kuin koskaan ennen. Ohjelmistoja kuten Open Office (nyk. Libre Office), Mozilla Firefox ja Linux-jakelut otetaan käyttöön yhä enemmän niin yksilö- kuin yritystasolla. Avoimen lähdekoodin periaate sisältää monia asioita, joita suljetulla lähdekoodilla on mahdotonta saavuttaa. Konkreettisten hyötyjen lisäksi avoimeen lähdekoodiin kuuluu ideologisia näkökantoja.

Avoimen lähdekoodin käyttöön liittyy paljon epäluuloa ja väärinkäsityksiä. Erityisesti yritysmaailmassa OSS:ään siirtymistä pidetään kummallisena ja riskialttiina asiana. Tämän takia tarkoituksenani on poistaa tutkielman keinoin epäluuloa avointa lähdekoodia kohtaan. Pyrin tutkielmassani selvittämään, mitä avoimella lähdekoodilla tarkoitetaan sekä mitä hyötyjä ja haittoja avoimeen lähdekoodiin liittyy. Esimerkkeinä OSS:stä käytän Linuxia ja sen variantteja. Tavanomaista Linux – Windows -vertailua olen pyrkinyt välttämään objektiivisuuden säilyttämiseksi, mutta olen ottanut mukaan muutamia vertailuja näiden järjestelmien kesken, koska Windows toimii hyvänä vertauspohjana. Olen pyrkinyt rajaamaan aineistoa siten, että vain olennaisimmat seikat käydään läpi.

2. Avoimen ja suljetun lähdekoodin raja

Avoimen lähdekoodin ohjelmisto julkaistaan useimmiten lisenssillä kuten GNU General Public License (GNU GPL). Vaikka lisenssejä on olemassa useita, voi-

daan niistä tehdä muutamia yleistyksiä yhteneväisyyksien perusteella, jolloin saadaan muodostettua yleiset avoimen lähdekoodin määritteet [OSI, 2011]:

1. Vapaa ohjelmiston jakelu
2. Lähdekoodin saatavuus
3. Vapaus käyttää lähdekoodia ohjelmiston muokkaamiseen ja johdannaisten luomiseen
4. Lisenssi ei saa syrjiä ihmisiä tai toimialoja
5. Lisensoidusta ohjelmistosta erotetulle yksittäiselle ohjelmalle tai ohjelmistosta tehdylle johdannaiselle pätevät alkuperäisen ohjelmiston lisenssin oikeudet
6. Lisenssi ei saa rajoittaa ohjelmiston mukana jaettua muuta ohjelmistoa
7. Lisenssi pitää olla teknologiasta riippumaton.

Listasta käy ilmi, että alkuperäisestä muokattu tai muuten johdettu ohjelmisto perii alkuperäisen ohjelmiston kantaman lisenssin, jolloin myös näiden ohjelmistojen lähdekoodi pitää olla julkisesti saatavilla ja muokattavissa. Yleinen väärinkäsitys on, että avoimen lähdekoodin ohjelmat ovat kaikki ilmaisia. Lista ei suinkaan kiellä ohjelmistosta maksamista: vapaa ohjelmiston jakelu tarkoittaa, että lisensoidusta tuotteesta voi periä maksua, mutta lisenssin käytöstä ei peritä maksua. Tärkeintä avoimen lähdekoodin periaatteessa on, että lähdekoodi on saatavilla, jolloin ohjelmaa voi muokata. Lisäksi lisenssi sallii ohjelmiston muokkaamisen ja muokatun ohjelmiston jakelun.

Suljetun lähdekoodin ohjelmistot käyttävät yleensä erityisiä tuotekohtaisia lisenssejä, mutta on olemassa muutamia valmiita lisenssejä [CC, 2011], joita voi hyödyntää ohjelmiston julkaisemisessa. Olennaista suljetun lähdekoodin periaatteessa on, että lähdekoodi ei ole vapaasti saatavilla. Riippuen lisenssistä ohjelmistoa voi saada kopioida ja käyttää kaupalliseen tarkoitukseen, mutta ohjelmiston muokkaaminen on joka tapauksessa mahdotonta. Esimerkkejä suljetun lähdekoodin ohjelmistoista ovat Microsoft Windows -järjestelmät ja Apple Mac OS -järjestelmät. Tosin Mac OS -järjestelmissä on myös avoimen lähdekoodin komponentteja [Apple, 2011]. Näiden komponenttien lähdekoodin voi ladata ja ehdottaa muutoksia koodiin Applelle, joten koodin muokattavuus on varsin rajoitettua. Windows-järjestelmille voi kirjoittaa maksutta kolmannen osapuolen sovelluksia, mutta Mac OS -järjestelmät ovat tavallaan *suljetumpia* kuin Windows-järjestelmät: jotta Mac OS -yhteensopivia sovelluksia voi kirjoittaa, pitää ensin ostaa ohjelmistokehittäjälisenssi [Apple, 2011]. Toisin kuin Mic-

rosoft, Apple valvoo App Store -sovelluksen kautta julkaistuja kolmannen osapuolen ohjelmistoja. Näin ollen kaikkia Applen tuotteille kehitettyjä ohjelmistoja ei välttämättä julkaista, jos nämä ohjelmistot eivät kunnioita Applen filosofiaa.

Applen kehittämä Mac OS -käyttöjärjestelmä on avoimen ja suljetun lähdekoodin rajamaastossa herättäen kysymyksen, missä avoimen ja suljetun lähdekoodin raja kulkee? Jos lähtökohdaksi otetaan edellä mainittu lista avoimen lähdekoodin lisenssien yhtäläisyyksistä, voidaan hahmottaa tietyt periaatteet, joita avoimen lähdekoodin ohjelman pitäisi noudattaa. On syytä kuitenkin muistaa, että vaikka tällaiset periaatteet täyttyisivät, ohjelmistokehittäjän filosofia ja toiminta voi rajoittaa ohjelmiston avoimuutta: Voidaan ajatella, että yhtiö julkaisee avoimen lähdekoodin järjestelmän, joka noudattaa yleisiä avoimen lähdekoodin periaatteita. Kuitenkin yhtiö valvoo tiukasti järjestelmänsä kanssa yhteensopivien ohjelmistojen julkaisua päättäen suvereenisti, mikä ohjelmisto pääsee julkiseen levitykseen.

Linuxilla tarkoitetaan avoimen lähdekoodin käyttöjärjestelmäydintä, joka suorittaa käyttöjärjestelmän perustoimintoja. Näitä perustoimintoja ovat esimerkiksi laitteiston tunnistaminen ja ohjelmien lataaminen käynnistysvaiheessa. Esimerkki avoimen lähdekoodin käyttöjärjestelmästä on Linuxiin perustuva Ubuntu, josta suurin osa julkaistaan GNU GPL -lisenssillä. Ubuntussa on myös joitain ohjelmistokomponentteja, esimerkiksi osa mediakoodekeista, joiden käyttöä on rajoitettu. Näistä riippumatta Ubuntun perusversio, eli versio, jonka voi ladata ohjelmistokehittäjän sivuilta, on avoin ja maksuton. Lisäksi suuri osa Ubuntulle julkaistavista ohjelmista on maksuttomia ja avoimia.

3. Linuxin menestyminen

Jos tarkastellaan kuluttajamarkkinoita, niin Linux-jakelujen levinneisyyttä on vaikea mitata johtuen tavasta, jolla jakeluja jaellaan: yleensä jakelu ladataan internetistä jakelun kehittäjän sivuilta. Latauskerrat eivät kuitenkaan kerro koko totuutta, sillä samaa ladattua jakelua voidaan käyttää – tai olla käyttämättä – uudelleen usean eri käyttäjän useassa eri koneessa. Jakeluja jaellaan myös dvd-levyillä ja torrenteina, mikä lisää käyttäjien lukumäärän laskemisen hankaluutta. Näin ollen kaikista käytössä olevista käyttöjärjestelmistä Linuxin arvioitu osuus on vuonna 2011 noin viisi prosenttia [W3Schools, 2011]. Huomattavaa on, että arvio perustuu W3Schools-sivuston lokitiedostoihin, joten kyseessä on ainoastaan suuntaa antava arvio. Silti arvio viittaa siihen, että Linux-käyttäjien määrä on suhteellisen pieni.

Toisaalta Linuxia käytetään hyvin laajasti erityiskäyttötarkoituksiin ja organisaatioiden käyttötarkoituksiin. Viidestäsadasta nopeimmasta supertietoko-

neesta 82,6% toimi kesäkuussa 2011 Linuxiin perustuvalla käyttöjärjestelmällä [TOP500, 2011]. Lisäksi Linux on suosittu käyttöjärjestelmä organisaatioiden palvelimissa ja web-palvelimissa [W3Techs, 2011]. Linuxin suosio on organisaatioissa ja erityiskäyttötarkoituksissa suhteellisen suuri verrattuna kuluttajamarkkinoihin, mutta tällainen tietylle markkina-alueelle kohdistuva suosio on odotettua; Linux on muokattavuutensa ansiosta monikäyttöinen rakennusalusta.

4. Avoimen lähdekoodin periaatteen hyödyt ja haasteet

Useat eri yritykset ovat vaihtaneet suljetun lähdekoodin ohjelmistoja avoimen lähdekoodin ohjelmistoihin ja säästäneet tällä tavoin erilaisissa kuluissa, kuten ohjelmistojen ostohinnoissa, lisenssimaksuissa ja ohjelmistopäivityksissä. Esimerkiksi Amazon.com pystyi leikkaamaan teknologiainfrastruktuurinsa kuluja 71 miljoonasta dollarista 54 miljoonaan dollariin vaihtamalla osan järjestelmistään avoimen lähdekoodin järjestelmiin [Nagy et al., 2010]. Ottamalla käyttöön OS-ohjelmistoja yritys pystyy siis saavuttamaan merkittäviä kuluvähennyksiä. Kuluvähennyksiin vaikuttavia tekijöitä voidaan selittää vetoamalla loogisiin ja yleisiin tekijöihin, kuten rajoittamaton käyttö ja tehokkuus, jotka tulevat esille OS-ohjelmistojen käyttämisessä, vaikka Amazon.com-tapauksen yksityiskohdat ovat salaisia. Avoimen lähdekoodin lisenssien yhteneväisyyksistä voidaan johtaa OSS-ominaisuuksia, kuten vapaa muokattavuus ja vapaa ohjelmiston uudelleenjako. Lisäksi OS-ohjelmistot ovat periaatteellisesti kevyitä, mikä tarjoaa ohjelmistojen käyttäjälle mahdollisuuden joko ottaa uudelleen käyttöön vanhemman teknologian laitteistoja tai jatkaa tämänkaltaisten laitteistojen käyttämistä. Ottaen huomioon OSS:n alhaisen käyttöasteen herää kysymys, miksi yritykset jatkavat suljetun lähdekoodin ohjelmistojen käyttämistä.

4.1 Avoimen lähdekoodin hyödyt

Avoimen lähdekoodin ohjelmistojen käyttämisessä on monia hyötyjä, joita käsitellään seuraavaksi. Vahvuuksista kirjoitettaessa esimerkkinä käytetään paljon Linuxia useista syistä. Se on esimerkillinen avoimen lähdekoodin ohjelmisto, minkä lisäksi sen saanut suosio ja kirjoittajan oma Linux-osaaminen puoltavat sen valitsemista.

4.1.1 Kustannusleikkaukset

Kuten jo aiemmin mainittiin, OSS:ää käyttämällä yritykset pystyvät vähentämään merkittävästi teknologiakustannuksiaan. Kustannusvähennyksiin vaikuttaa ensinnäkin lisenssimaksun puuttuminen. Vaikka OSS-kehittäjä voi määrätä ohjelmistolle hinnan, lisensoidun tuotteen loppuhinta koostuu vain ohjelmiston

myyntihinnasta. Esimerkiksi Windows 7 Ultimate 32-bitin hinta, noin 210 euroa, on osuudeltaan noin neljännes 800 euron pöytätietokoneen kokonaishinnasta [Verkkokauppa.com, 2011]. Vastaava OS-käyttöjärjestelmä on esimerkiksi Ubuntu 11.10, jonka hinta on noin 5,90 euroa, jos jakelun tilaa dvd-levylle poltettuna [Canonical, 2011]. Muuten Ubuntun voi ladata ilmaiseksi internetistä. Tietysti Windows 7:stä on olemassa halvempiakin versioita, mutta esimerkkiin valittiin kallein versio, sillä Ubuntusta on olemassa vain täysi versio, joten oikeanlaisen vertailupohjan vuoksi täytyi valita myös Windows 7:n täysi versio. Nimellishinnan lisäksi OSS-lisenssit, kuten GNU GPL, ovat perimmiltään sallivia lisenssejä. Tämä tarkoittaa sitä, että lisenssit antavat enemmän valtaa ohjelmiston käyttäjälle kuin toisinpäin, eli ne suojelevat ohjelmistoa käyttäjältä. Lisenssien suoma oikeus muokata lähdekoodia vapaasti johtaa mahdollisuuteen kustomoida järjestelmiä eri käyttötarkoituksia varten, jolloin voidaan suhteellisen lyhyessä ajassa luoda erikoistilanteisiin soveltuvia järjestelmiä hyvin vähäisillä kustannuksilla. Lisäksi säästöjä syntyy siitä, että OSS – erityisesti Linux – on usein resurssien käytöltään ja kooltaan optimoitu. Minimivaatimus keskusmuistin määräksi Ubuntu 11.10:lle on 64Mt ja kovalevyn kooksi suositellaan 5Gt, mikä mahdollistaa tämän hetken ladatuimman Linux-jakelun täyden version käyttämisen useita vuosia vanhoissa tietokoneissa [Ubuntu, 2011]. Vastaavasti Windows 7:n minimivaatimukset keskusmuistin määrälle on 1Gt ja kovalevyn koolle 16Gt [Windows, 2011]. Siispä kustannuksia voidaan vähentää Linuxia käytettäessä, koska järjestelmän käyttämiseksi ei tarvitse ostaa uutta tietokonetta. Tästä esimerkkinä mainittakoon se, että Linuxia on pystytty käyttämään jopa i486-tietokoneella [Kshetri, 2004].

4.1.2 Linuxin turvallisuus

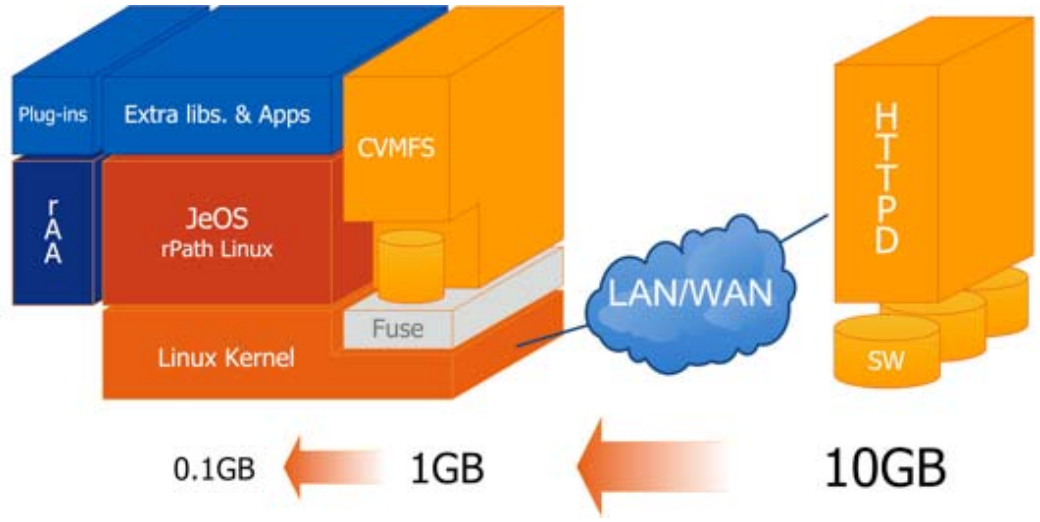
Turvallisuus on yksi Linuxin suurimpia vahvuuksia. Verrattuna Windowsiin Linuxiin kohdistettuja viruksia, haittaohjelmia ja matoja on esiintynyt harvoin [Rautiainen, 2001]. Syy hyökkäysten vähäiselle määrälle on monitahoinen, mutta yksi vaikuttavimmista tekijöistä on se, että hyökkäysten määrä on suhteessa käyttöjärjestelmän käyttäjien määrään. Kuitenkin Linuxin turvallisuutta voidaan perustella järjestelmäsuunnitteluteknisillä valinnoilla, kuten ytimen ja käyttäjän erotus, oikeudet tiedostojen käyttämiseen, OSS-monimuotoisuus ja lähdekoodin vapaa tarkastelu. Linuxissa ytimen käyttämät prosessit, tiedostot (kernel space), käyttäjän käyttämät prosessit sekä tiedostot (user space) ovat erotettu toisistaan, jolloin käyttäjällä ei ole suoraa oikeutta suorittaa matalan tason prosesseja, joiden väärinkäyttö vaarantaa järjestelmän toiminnan. Käyttäjän ja ytimen välillä portinvartijan roolissa toimii sudo-ohjelma (Superuser Do), jonka avulla käyttäjä voi suorittaa matalan tason prosesseja. Sudo-ohjelma vaa-

tii käyttäjältä tämän asettamaa salasanaa, jonka syötettyään käyttäjälle annetaan superuser-oikeudet eli järjestelmäylläpitäjän oikeudet. Näillä oikeuksilla käyttäjällä on valta suorittaa mitä tahansa prosesseja siten, että kaikki järjestelmän asettamat varotoimenpiteet ohitetaan. Linuxiin kohdistuva hyökkäys, esimerkiksi mato, on oletusarvoisesti estetty, sillä madolla ei ole oikeuksia hyväksikäyttää tai väärinkäyttää järjestelmäkriittisiä prosesseja. Ainoastaan käyttäjä voi antaa madolle oikeudet järjestelmän saastuttamiseen. Toinen Linuxin turvallisuutta edistävä asia on tiedostojen suorittamisoikeudet. Oletuksena jokaisella tiedostolla on vain lukuoikeus, ellei käyttäjä itse muuta suorittamisoikeutta. Käyttäjä voi siis lukea järjestelmässä olevia tiedostoja, mutta niihin kirjoittaminen vaatii suorittamisoikeuden muutoksen. Tämän muutoksen voi tehdä suorittamalla esimerkiksi komennon chmod =rw, jolloin käyttäjä saa oikeuden muokata tiedostoa. Vastaavasti komento chmod =rx antaa käyttäjälle oikeuden suorittaa tiedosto, eli kyseessä on tällöin yleensä skripti tai ohjelmapikakuvake. Ajatellaan esimerkiksi tilannetta, jossa käyttäjä avaa haitallisen sähköpostin, jonka seurauksena järjestelmään latautuu haittaohjelma esimerkiksi sähköpostissa olevan kuvan kautta. Käyttäjän pitäisi sitten antaa chmod-komento haittaohjelmalle, jotta tämä pystyisi toimimaan. Lisäksi käyttäjän pitäisi antaa haittaohjelmalle sudo-komento, jos tämä yrittäisi suorittaa matalan tason prosesseja tai yrittää muokata tiedostoja user spacen ulkopuolella eli kotihakemiston ulkopuolisia tiedostoja. On epätodennäköistä, että käyttäjä tekisi näin tiedostamatta tekojensa vaikutusta. Kolmas turvallisuustekijä on OSS:n monimuotoisuus: Linux-jakeluja on useita, mikä tarkoittaa, että haittaohjelma, joka toimii Ubuntulla, ei välttämättä toimi esimerkiksi Sabayon Linuxilla. Tässä mielessä OSS:n monimuotoisuus suojelee Linuxia hyökkäyksiltä. Neljäs turvallisuutta parantava tekijä, vapaa lähdekoodin tarkastelu, johtuu OSS-periaatteesta ja toimii siten, että useampien ihmisten tarkastellessa lähdekoodia ohjelmistovirheet löytyvät nopeammin ja ovat helpompia korjata. Lopuksi on syytä muistuttaa, että heikoin lenkki turvallisuudessa on usein käyttäjä [Rautiainen, 2001].

4.1.3 Vapaa muokattavuus

Avoimen lähdekoodin periaatteesta johtuva tekijä, vapaa muokattavuus, edistää OSS:n käyttöönottamista, koska lähdekoodia voidaan muokata mille tahansa alustalle sopivaksi. Linuxia käytetään tällä hetkellä useilla eri alustoilla jääkaapeista hiukkaskiihdyttimiin, mikä kertoo Linuxin muokattavuudesta. Avoimen lähdekoodin periaatteen mukaan lähdekoodin voi esimerkiksi ladata internetistä ja sitä voi vapaasti muokata, jolloin rajoja ohjelman kehittämiselle ei ole. Kehitettävää ohjelmaa ei tarvitse koodata alusta asti, vaan pohjana voidaan

käyttää jotain valmista järjestelmää, kuten Linux. Uusi ohjelma voidaan kehittää Linux-jakelun päälle, jolloin jakelu säilyy alkuperäisenä tai vaihtoehtoisesti uusi ohjelma voidaan kehittää siten, että jakelusta poistetaan ja muokataan moduuleita, jolloin jakelusta syntyy kokonaan eri ohjelma. Esimerkiksi CernVM on Linuxiin perustuva käyttöjärjestelmä hiukkaskiihdyttimille ja nimensä mukaan järjestelmää kehittää European Organization for Nuclear Research (CERN) [Buncic et al., 2010]. CernVM-järjestelmää voidaan käyttää alustasta riippumatta virtualisoinnin kautta, mikä tarkoittaa, että CernVM toimii niin Linuxilla, Windowsilla ja Mac OS:llä. Kuva 1 havainnollistaa asiaa.



Kuva 1. CernVM- järjestelmän rakenne.

Kuvasta nähdään, että Linuxia käytetään koko systeemin perustana ja perustoiminnallisuuden tarjoajana. Perustoiminnallisuudella tarkoitetaan tässä sellaisia toiminnallisuuksia kuten laitteiston tunnistus, ajurien asennus ja tuki eri tiedostojärjestelmille. Lisäksi kuvasta nähdään, että CernVM käyttää rPath Linuxia, joka on johdannaisten luomiseen hyvin soveltuva jakelu ja on binäärisesti yhteensopiva Scientific Linuxin kanssa [rPath Linux, 2011]. Tämä mahdollistaa ohjelmistokomponenttien hyväksikäytön: ”we can reuse the software packages already built and tested in that environment (Scientific Linux) and simply run them unmodified --” [Buncic et al., 2010]. Modulaarisuuden avulla ohjelmistokomponentteja voidaan saumattomasti soveltaa CernVM:ään. Näin ollen modulaarisuus omalta osaltaan lisää Linuxin muokattavuutta, eikä järjestelmän muokkaaminen vaadi ohjelmointitaitoja, mikä puolestaan nopeuttaa muokkaamista merkittävästi. Yksityiskohtiin menemättä voidaan siis todeta, että Linux – kuten OSS yleensä – on erittäin monipuolinen järjestelmä, joka tarjoaa joko kehitysalustan, käyttöjärjestelmän tai molemmat tarpeista riippuen.

4.1.4 Modulaarisuus

Modulaarisuus on OSS:lle yksi keskeisimmistä OSS:n menestymiseen ja jatkumiseen vaikuttavista tekijöistä. Modulaarisuudella tarkoitetaan yleisesti ohjelmiston jakamista ohjelmistopaketteihin tai moduuleihin, joita voidaan lisätä tai poistaa. Moduulien lisäämisellä voidaan lisätä järjestelmään toiminnallisuutta esimerkiksi Ubuntussa usbhid-moduulin lisääminen mahdollistaa usb-laitteiden tuen. Vastaavasti usbhid-moduulin poistaminen johtaa usb-laitetuen poistumiseen. Järjestelmään voidaan lisätä tai siitä voidaan poistaa moduuleita ajonaikaisesti, mikä nopeuttaa ja helpottaa järjestelmän konfigurointia. Linux tarvitsee harvoin uudelleenkäynnistämistä uusien ajureiden tai ohjelmien asennuksen yhteydessä, joten Linux sopii erinomaisesti ympäristöön, jossa maksimaalinen järjestelmän käytettävyysaika (uptime) on olennaista. Teknisen hyödyn lisäksi OS-projektit hyötyvät modulaarisuuden noudattamisesta, koska kehittäjät voivat keskittyä hallitsemaan ja koodaamaan pienempiä järjestelmän osia ja lisäksi yhtä moduulia kehittämässä voi olla samanaikaisesti monta koodaria. Tästä johtuen OS-projektit tarvitsevat vähemmän koordinaatiota, jolloin projektin hallitseminen helpottuu ja todennäköisesti projektin lopputulos paranee.

4.2 Avoimen lähdekoodin haasteet

Avoimen lähdekoodin ohjelmien käyttö yrityksissä on tämän tutkielman kirjoittamisen aikaan markkinaosuudeltaan vähemmistöä. On olemassa monia tekijöitä, jotka haittaavat OSS:n käyttöönoton leviämistä ja seuraavaksi tarkastellaan tällaisia leviämistä estäviä tekijöitä. Vaikka tarkastelussa on OSS:n kohtaamat ongelmat, sanalla haasteet on tarkoitus kuvastaa OSS:n selviämistä näistä ongelmista. Taulukko 1 havainnollistaa OSS:n kohtaamia haasteita ja jokaiselle haasteelle on ehdotettu mahdollinen ratkaisu. Taulukkoa tutkimalla saadaan selville ainakin osa käyttöönoton leviämistä estävistä tekijöistä.

Este	Kuvaus	Ehdotettu ratkaisu
Tiedon puute	Ei ole tietoa ohjelmistojen saatavuudesta tai relevanssista, ei ole teknistä osaamista ohjelmiston käyttämiseksi tai taloustietoa ohjelmiston kustomoimiseksi	Avoimen lähdekoodin kirjastojen tarkkailu, omien työntekijöiden kouluttaminen ja järjestelmän toteutuksen ja ylläpidon ulkoistaminen
Perinnejärjestelmien integroiminen	Kykenemättömyys kommunikoida muiden perinnejärjestelmien kanssa	Väliohjelmistojen käyttäminen
Pirstaloituminen	Useat eri tahot kehittävät OSS:ää, jolloin ohjelmistot eivät välttämättä ole yhteensopivia keskenään	Ongelma ratkeaa itsestään ajan myötä, kun itsenäiset OSS-standardit kehittyvät
Uponneet kustannukset	Suljetun lähdekoodin ohjelmistoihin on aiemmin tehty sijoituksia, joita on mahdoton saada hyvitettyä	OSS:n käyttäminen sellaisissa tehtävissä, joissa suljettua lähdekoodia ei ole vielä hyödynnetty. OSS:n käytöstä saatavien tulojen vertaaminen suljetun lähdekoodin ylläpitokustannuksiin
Tekninen kehittymättömyys	Ohjelmistot eivät ole ammattilaisten kehittämiä ja tuen saaminen vaihtelee	Avoimelle lähdekoodille on kehitetty kypsyysmalleja ja lisäksi erinäiset tahot tarkkailevat avoimen lähdekoodin tilaa

Taulukko 1. Avoimen lähdekoodin kohtaamat esteet ja ehdotetut ratkaisut.

4.2.1 Tiedon puute

Ensimmäinen käyttöönoton este on tiedon puute. Yrityksien teknologiasta vastuussa olevat henkilöt voivat olla tietoisia OSS:n olemassaolosta ja merkityksestä, mutta heillä ei ole kuitenkaan riittävää tietoa avoimen lähdekoodin vastaavista ohjelmistoista, OSS:n relevanssista tai teknisen tason vaatimuksista. Tietysti yritykset voivat hankkia teknistä osaamista rekrytoimalla ammattilaisia, jotka osaavat valita oikeat OS-tuotteet ja kustomoida ne yrityksen käyttöön sopiviksi, mutta kyseessä on rekrytoinnista päättävien henkilöiden riittämätön tieto. Lisäksi yrityksen taloushallinnolle pitää pystyä osoittamaan, että OS-

ohjelmisto on yhtä luotettava, tuettu ja helppokäyttöinen kuin vastaava kaupallinen suljetun lähdekoodin ohjelmisto. Pelkästään ohjelmiston ilmaisuus on itsessään heikko perustelu vaihtaa olemassa olevat ohjelmistot toisiin ohjelmistoihin, sillä taloushallinto usein tarvitsee konkreettisia esimerkkejä vastaavien ohjelmistojen käyttöönoton hyödyistä, haitoista ja piilevistä kustannuksista. Kysymyksiin, kuten ”voiko ilmainen ohjelma olla luotettava?”, ”kuinka menestynyt tällainen ohjelma on?” ja ”onko tällaista ohjelmaa helppo käyttää?”, on pystyttävä vastaamaan perustellusti. Ongelmana on, että menestystarinoita OS-ohjelmistoihin siirtymisestä saaduista hyödyistä on toistaiseksi liian vähän. Tilastotietoa OSS:n hyödyistä ja haitoista on liian vähän tai tilastot eivät perustu luotettaviin lähteisiin, mikä johtaa epävarmuuteen OSS:n käyttöönoton suhteen. Avoimen lähdekoodin ohjelmistojen ilmaisuus johtaa taloudellisten resurssien niukkuuteen, joka estää laajamittaisen markkinoinnin, ja näin tiedonpuutetta ja epävarmuutta ei pystytä vähentämään. Taulukossa ehdotettua ratkaisua pitää siis täydentää lisäämällä siihen ”markkinointihype”: OSS-yhtiöiden ja yhteisöjen pitää mainostaa OSS:ää ja erityisesti Amazon.comin kaltaisia tapauksia pitää tuoda esille vertailupohjana OSS:tä saatuihin hyötyihin ja vaarattomuuteen.

4.2.2 Perinnejärjestelmien uusiminen

Toinen käyttöönoton este on pitkään käytössä olleiden tietojärjestelmien eli niin sanottujen perinnejärjestelmien vaihtaminen avoimen lähdekoodin järjestelmiin. Tällaiset järjestelmät on kehitetty jo aikaisessa vaiheessa suorittamaan yhtiön toiminnan kannalta kriittisiä tehtäviä, esimerkiksi tuotantotehtäviä, jolloin yritykset ovat haluttomia vaihtamaan jopa vuosikymmeniä vanhoja järjestelmiä uusiin järjestelmiin. Perinnejärjestelmät on kehitetty käyttäen vanhentuneita tekniikoita ja suunnitteluperiaatteita, joten näiden järjestelmien päivittäminen uusilla ohjelmistokomponenteilla on yhä hankalampaa yhteensopivuusongelmista johtuen. Yhteensopivuusongelmia ovat tässä kohden esimerkiksi järjestelmän kykenemättömyys viestiä muiden järjestelmien kanssa tai tiedontallentamismenetelmien vanhanaikaisuus. Lisäksi perinnejärjestelmät ovat pitkälti yrityskohtaisia ja ajan kuluessa järjestelmien käyttäjien määrä laskee, mikä johtaa lopulta tilanteeseen, jossa yritys ei saa koulutusta tai tukea käyttämälleen järjestelmälle. Yritys joutuu siis kouluttamaan itse uudet työntekijänsä siten, että järjestelmän käytön oppineet työntekijät kouluttavat uusia työntekijöitä, mikä puolestaan on työnteon kannalta tehotonta. Ongelmallisuutta lisää se, että uusia työntekijöitä voi olla vaikeata rekrytoida sellaiseen työhön, joka perustuu ”jäännejärjestelmien” käyttämiseen. Ratkaisuna perinnejärjestelmien päivittämiseen on avoimen lähdekoodin väliohjelmistojen käyttö.

Väliohjelmistot tarjoavat eräänlaisen rajapinnan erillisten järjestelmien välille antaen järjestelmille mahdollisuuden kommunikoida keskenään väliohjelmistojen kautta. Merkittävä etu väliohjelmistojen käytössä on se, että niiden tarjoaman rajapinnan avulla perinnejärjestelmän ylläpito helpottuu.

4.2.3 Avoimen lähdekoodin ohjelmistojen pirstaloituminen

Kolmas este käyttöönotolle on *forking*, jonka suomenkielinen vastine voisi olla pirstaloituminen tai monimuotoisuus. Tämä termi on suhteellisen uusi mediassa, joten virallista suomenkielistä termiä ei esimerkiksi Tietotekniikan liitto ry:n (TTL) sanakirjasta tai Wikipediasta löytynyt. Pirstaloituminen ei ole uusi ilmiönä – eikä terminä tietotekniikan maailmassa – avoimen lähdekoodin ohjelmistojen evoluutiossa, vaan yksi varhaisimmista pirstaloitumisista on Berkeley Software Distribution (BSD)-käyttöjärjestelmän kehityksen alkaminen vuonna 1977. Kehitys lähti liikkeelle AT&T:n Bell Laboratoriesin julkaistua UNIX-järjestelmänsä, jonka lähdekoodin pohjalle BSD perustuu. Pirstaloitumisella tarkoitetaan yksinkertaisimmillaan siis lähdekoodin kopioimista ja muokkaamista uuden ohjelmiston ominaisuuksien toteuttamiseksi. Pirstaloituminen on nykyään kehittynyt kokonaisvaltaisemmaksi prosessiksi kuin mitä lähdekoodin kopioimisella ja muokkaamisella voidaan tarkoittaa: lähdekoodi kopioidaan ainoastaan uuden järjestelmän rakentamisen perustaksi, jonka päälle järjestelmä kehitetään, jolloin alkuperäinen lähdekoodi toisesta järjestelmästä käsittää vain osan uuden järjestelmän koodista. Tällä tavoin syntyy uusi järjestelmä toisen järjestelmän pohjalta ja on luonnollista puhua tässä yhteydessä OSS:n evoluutiosta, koska pirstaloitumisessa on myös kyse ympäristötekijöiden vaikutuksesta. Ympäristötekijät voivat koostua kehittäjien erimielisyyksistä, lakiteknisistä seikoista, kuten lisenssin rajoittavuudesta tai alkuperäisen ohjelmistoprojektin kehityksen loppumisesta. Avoimen lähdekoodin evoluution kautta pirstaloitumisesta muovautuu monimuotoisuus, jossa yksi järjestelmä on kymmenien järjestelmien esi-isä. Toisaalta pirstaloitumisesta puhuminen on korrektia, kun tarkastellaan OSS:n monimuotoisuuden negatiivisia vaikutuksia. Ongelmana pirstaloitumisessa on eri avoimen lähdekoodin järjestelmien epäjohdonmukaisuus, jota voi havainnollistaa Android-mobiilikäyttöjärjestelmän graafisen käyttöliittymän päälle laitevalmistajien rakentamilla graafisilla käyttöliittymäkomponenteilla. Androidia käyttävillä kännykkävalmistajilla kuten HTC:lla ja Samsungilla on toisistaan eroavat kustomoidut käyttöliittymät, mikä asettaa Androidin epäedulliseen asemaan: vaikka kahdessa eri älypuhelimessa on asennettuna sama Android-järjestelmäversio, käyttöliittymä toimii silti erilailla molemissa puhelimissa. Standardien puuttuminen Android-käyttöliittymäsuunnittelussa voi synnyttää käyttäjälle mielikuvia järjestelmän käyttämisen

vaikeudesta, kun useimmat ”androidit” toimivat erilailla. Lisäksi käyttäjälle voi jäädä epäselväksi, mikä Android on. Miten näissä molemmissa puhelimissa voi olla Android, kun molemmat näyttävät erilaisilta ja toimivat erilailla? Sama ongelmallisuus koskee myös Linuxia, sillä marraskuussa 2011 Linux-jakeluja on jaossa 312 kpl [DistroWatch, 2011]. Standardisoinnin puuttumisen lisäksi OSS:n käyttöönoton leviämistä hankaloittavat yhteensopivuusongelmat muiden järjestelmien kanssa. Erityisesti OSS:n yhteensopimattomuus suljetun lähdekoodin ohjelmien kanssa on ongelma, jonka yritykset kokevat vaikeana ongelmana, joka estää OSS:n käyttöönoton yrityksissä. Organisaatiot, kuten Linux Foundation ja Open Source Initiative, kehittävät ja valvovat OSS:n standardeja, joten ajan myötä yhteensopimattomuus todennäköisesti ratkeaa itsestään OSS-kehittäjien seuratessa standardeja. Tämä ei välttämättä kuitenkaan riitä ongelman ratkaisemiseksi, sillä vaikka avoimen lähdekoodin ohjelmistot hyötyvät yhteensopivuudesta suljetun lähdekoodin ohjelmistojen kanssa, suljetun lähdekoodin ohjelmistot hyötyvät yhteensopimattomuudesta avoimen lähdekoodin ohjelmistojen kanssa [Cheng et al., 2011]. Näin ollen suljetun lähdekoodin kehittäjien liiketoimintaa edistävä strategia olisi koodata ohjelmiaan siten, että OSS-kehittäjien olisi mahdollisimman vaikeata saavuttaa yhteensopivuutta näiden ohjelmien kanssa.

4.2.4 Uponneet kustannukset

Neljäs este on uponneet kustannukset, joilla tarkoitetaan sellaisia kokonaiskustannuksia, joita ei voida tulevaisuudessa hyvittää. Esimerkiksi yrityksen siirtyminen avoimen lähdekoodin ohjelmistojen käyttämiseen johtaisi uponneisiin kustannuksiin, koska aikaisemmin käytettyihin suljetun lähdekoodin ohjelmistoihin käytettyjä investointeja ei pystyttäisi enää hyvittämään. Siispä OSS:ään siirtyminen vaikuttaa kestämättömältä ajatukselta uponneiden kustannusten valossa. Yritysten pitäisi verrata OSS:ään siirtymisestä saatuja kustannusleikkauksia suljetun lähdekoodin ohjelmistojen aiheuttamiin kustannuksiin tai vaihtoehtoisesti yritykset voisivat ottaa käyttöön avoimen lähdekoodin ohjelmistoja niille alueille, joilla suljetun lähdekoodin ohjelmistoja ei käytetä. Joka tapauksessa uponneet kustannukset ovat käyttöönoton estämisessä merkityksettään muita esiteltyjä tekijöitä vähäisempiä.

4.2.5 Teknologinen kehittymättömyys

Viimeisenä esteenä on avoimen lähdekoodin teknologinen kehittymättömyys. Kyseessä on kuitenkin yleinen harhaluulo, jota suljetun lähdekoodin kehittäjien markkinointipuheet pitävät elossa [Nagy et al., 2010]. On mahdollista, että OS-projektin kehittäjä voi halutessaan lopettaa työskentelyn projektin parissa, sillä

OS-projektien kehitys on joustavaa ja modulaarista ja kehittäjät tekevät työtä usein ilmaiseksi. Kehittäjillä ei näin ollen ole erityisiä velvoitteita, kuten työsuhde ja asiakassopimus, saada projektia valmiiksi. Kuitenkin avoimen lähdekoodin projektit, kuten Linux, Apache, Mozilla Firefox ja MySQL, osoittavat kehitysmallin toimivuuden sekä teknologisen osaamisen tason.

4.2.6 Tukipalvelut ja ohjelmistoportfolion niukkuus

Edellä mainittujen esteiden lisäksi on syytä mainita vielä kaksi muuta tekijää: ohjelmiston asennuksen jälkeinen tukipalvelu ja asennetun ohjelmiston relevantti ohjelmistoportfolio. Otetaan esimerkiksi OpenSuse, joka asennetaan yrityksen tietokoneelle. Oletetaan sitten, että asennuksen jälkeen järjestelmä ei pysty kommunikoimaan tietokoneen wlan-verkkosovittimen kanssa. Ainoa tapa saada ilmaista ja välitöntä tukea kuvattuun ongelmaan on kysyä apua kehittäjäyhteisön foorumilla, joten avun saaminen on riippuvainen täysin muiden foorumin käyttäjien teknisestä osaamisesta ja halukkuudesta auttaa. Toinen mahdollisuus on ilmoittaa kehittäjille ongelmasta ohjelmistovirheenä, jolloin kehittäjät tutkivat asiaa ja mahdollisesti julkaisevat ongelman korjaavan päivityksen. Kuitenkaan varsinaista tukipalvelua ei usein ole tarjolla. Maksullista tukea on saatavilla muutamilta tahoilta, kuten Red Hat ja United Linux, mutta usein yritykset ovat haluttomia maksamaan tuesta, kun ohjelmisto muuten on ilmainen [Suman and Bhardwaj, 2003]. Yrityksille pitää tähdentää, että juuri tällainen maksullinen palvelu on Linux-kehittäjien ansaintamalli (revenue model). Tuen saamisen lisäksi ohjelmistoportfolion laajuus on este useiden OS-ohjelmistojen käyttöönotolle. Esimerkiksi Linuxin kanssa yhteensopivia ohjelmia on vähemmän kuin Windowsin kanssa yhteensopivia. Suljetun lähdekoodin ohjelmia vastaavia avoimen lähdekoodin ohjelmia pitää olla enemmän, jotta yritykset voidaan vakuuttaa OSS:ään siirtymisestä saaduista hyödyistä.

5. Yhteenveto ja keskustelua

Avoimella lähdekoodilla on ainutlaatuisia vahvuuksia suljettuun lähdekoodiin verrattuna, kuten vapaa muokattavuus, kustannusten aleneminen ja ideologiset ominaisuudet. Useat hyvin menestyneet OS-projektit ovat osoittaneet avoimen lähdekoodin periaatteen toimivuuden ja näiden projektien menestyminen takaa OSS-kehityksen jatkumisen. Menestymisen kasvua kuitenkin hidastavat OSS:n kohtaamat haasteet, kuten tiedon puute, standardien puuttuminen, tukipalvelun puutteellisuus ja ohjelmistoportfolion suppeus. Nämä haasteet muodostavat esteen OSS:n käyttöönoton leviämiselle, niin yrityksissä kuin kuluttajakäytössä. Voitettuaan nämä esteet OSS voi horjuttaa nykyistä suljetun lähdekoodin lähes monopolistista asemaa ja muuttaa tietotekniikan periaatteita, käsityksiä

sekä arvoja. Tulevaisuudessa avoin lähdekoodi voidaan nähdä perusperiaatteena kaikelle tietotekniselle kehitykselle. Ohjelmistokehitys edistää tasa-arvoisuutta ollessaan vapaata ja läpinäkyvää: lähdekoodin avoimuus poistaa tekniikoiden tahallisen syrjimisen, ihmisten yksityiselämän vakoilemisen ja intellektuaalisen omaisuuden varastamisen. Näiden lisäksi avoimella lähdekoodilla on suuri potentiaali tehdä ohjelmistoista sellainen hyödyke, joka on taloudellisesti jokaisen ihmisen saatavilla. On perusteltua väittää, että nykyään tietotekniikan käyttäminen on yhtä olennainen osa lähes jokaisen ihmisen arkipäiväistä elämää kuin autolla liikkuminen. Yleensä kun henkilö on ostanut hyödykkeen, omistajanvaihdos antaa oikeuden käyttää hyödykettä henkilön haluamalla tavalla. Hyödykettä voidaan käyttää valmistajan tarkoittamalla tavalla, hyödykkeelle voidaan keksiä uusi käyttötapa tai hyödyke voidaan purkaa. Näin ollen auton purkaminen osiin on sallittua, mutta ohjelmiston osiin purkaminen on silti kiellettyä. Suljetun lähdekoodin ohjelmistojen omistajanvaihdossa piilee erikoinen ominaisuus, joka estää ohjelmistonkäyttäjää saamasta täyttä omistajuussuhdetta ohjelmistoon nähden. Huomioimalla aiemmin mainittu tietotekniikan rooli ihmisten elämässä tämä ominaisuus tekee suljetun lähdekoodin periaatteesta epähyväksyttävän ohjelmistokehityksessä. Avoin lähdekoodi puolestaan tukee omistajanvaihdosta johtuvien oikeuksien täyttymistä, joten ohjelmistonkäyttäjällä on oikeus käyttää ohjelmistoa parhaaksi näkemällään tavalla. Konkreettisesti tämä oikeuksien täysi siirtyminen näkyy siten, että esimerkiksi vuonna 2003 Intiassa noin 950 miljoonan ihmisen internetinkäytön esti käyttöjärjestelmän käyttöliittymän englanninkielisyys, jolloin OSS tarjoaa ratkaisun käyttöliittymän kääntämiseksi paikalliselle kielelle ja sen murteille [Suman and Bhardwaj, 2003]. On kuitenkin syytä pohtia, millä tavalla avoin lähdekoodi muuttaa tietotekniikan maailman ja ohjelmistomarkkinat. Onko avoin lähdekoodi ratkaisu ohjelmistoyritysten harjoittamaan häikäilemättömään kilpailuun, jossa unohdetaan käyttäjä ja hänen oikeutensa, vai aiheuttaako avoin lähdekoodi ohjelmistomarkkinoiden hajautumisen ja terveen kilpailun hajoamisen?

Viiteluettelo

[Apple, 2011] Apple Inc., Apple Developer Programs. http://developer.apple.com/programs/. Checked 17.10.2011.

[Apple, 2011] Apple Inc., Open at the Source. http://www.apple.com/open-source/. Checked 18.10.2011.

[Buncic et al., 2010] P. Buncic, C. Aguado Sanchez, J. Blomer, L. Franco, A. Harutyunian, P. Mato and Y. Yao, CernVM – A virtual software appliance for

LHC applications. In: *Proc. of 17th International Conference on Computing in High Energy and Nuclear Physics* , 1–10.

[Canonical, 2011] Canonical Ltd., Ubuntu Shop, http://shop.canonical.com/product_info.php?products_id=915.Checked 1.12.2011.

[CC, 2011] Creative Commons, Choose a License. http://creativecommons.org/choose/. Checked 17.10.2011.

[Cheng et al., 2011] Hsing Kenneth Cheng, Yipeng Liu and Qian (Candy) Tang, The impact of network externalities on the competition between open source and proprietary software. *Journal of Management Information Systems* **27**, 4 (Spring 2011), 201–230.

[DistroWatch, 2011] DistroWatch, DistroWatch Page Hit Ranking, http://distrowatch.com/dwres.php?resource=popularity. Checked 1.12.2011.

[Kshetri, 2004] Nir Kshetri, Economics of Linux adoption in developing countries. *IEEE Software* **24,** 1 (January/February 2004), 74–81.

[Nagy et al., 2010] Del Nagy, Areej M. Yassin, and Anol Bhattacherjee , Organizational adoption of open source software: barriers and remedies. *Communications of the ACM* **53,** 3 (March 2010), 148-151.

[OSI, 2011] The Open Source Initiative, The Open Source Definition. http://www.opensource.org/docs/osd. Checked 17.10.2011.

[Rautiainen, 2001] Sami Rautiainen, Travelling with Linux malware: Is Linux security for real? *Information Security Technical Report* **6**, 4 (2001) 58–64 .

[rPath Linux, 2011] rPath Linux documentation, rPath Linux, http://wiki.rpath.com/wiki/rPath_Linux. Checked 10.12.2011.

[Suman and Bhardwaj, 2003] Yogesh Suman and A. K. Bhardwaj , Open source software and growth of Linux: the Indian perspective. *DESIDOC Bulletin of Information Technology* **23**, 6 (November 2003), 9–16 .

[TOP500, 2011] TOP500, Operating System Share for 06/2011. http://top500.org/stats/list/37/os. Checked 21.10.2011.

[Ubuntu, 2011] Ubuntu Documentation, Meeting Minimum Hardware Requirements, https://help.ubuntu.com/11.10/installation-guide/hppa/minimum-hardware-reqts.html. Checked 18.11.2011.

[Windows, 2011] Windows, Windows 7 system requirements, http:// windows.microsoft.com/en-US/windows7/products/system-requirements. Checked 2.12.2011.

[W3Schools, 2011] W3Schools Internet Developers Portal, OS Platform Statistics. http://www.w3schools.com/browsers/browsers_os.asp. Checked 21.10.2011.

[W3Techs, 2011] World Wide Web Technology Surveys, Usage statistics and market share of Linux for websites. http://w3techs.com/technologies/details/os-linux/all/all. Checked 28.10.2011.

Virtuaalitaloudesta ja virtuaaliverotuksesta

Jude Laine

Tiivistelmä

Virtuaaliverotuksesta on puhuttu ensimmäisiä kertoja jo yli vuosikymmen sitten. Tässä tutkielmassa pyritään mahdollisimman tiiviisti mutta kattavasti tarkastelemaan virtuaalisten tavaroiden omistamista, virtuaalimaailmojen ympärille muodostunutta taloutta sekä virtuaaliverotusta.

Avainsanat ja -sanonnat: Virtuaaliverotus, verotus, virtuaalinen talous
CR-luokat: J.4, K.4.4

1. Johdanto

Verotus on nykyään arkipäivää jokaisessa kehittyneessä valtiossa, kuten ovat myös internet, PC- ja konsolipelit sekä käsite omistamisesta. Mutta vaikka kaikki nämä erillään ovat sinänsä helppoja ja kokemattomammillekin ymmärrettäviä asioita, ei niiden summa välttämättä ole läheskään yhtä itsestään selvä. Nykyään on useitakin pelejä, joita pelataan pelkästään internetin välityksellä. Parhaimpana ja tunnetuimpana esimerkkinä näistä ovat ehkä *massiiviset monipelattavat roolipelit* (massive multiplayer online role playing game, tai MMORPG) kuten *World of Warcraft* – tai tuttavallisemmin vain *WoW*, sekä todellista elämää ja maailmaa imitoiva *Second Life*, joka ei ole roolipeli mutta massiivimoninpeli joka tapauksessa.

Merkittävin ero tällaisissa peleissä yksinpelattaviin sisartuotteisiinsa on se, että pelaaminen ei tapahdu yksin, vaan sisältää sosiaalisesti suuriakin tosielämää muistuttavia elementtejä. Lisäksi tällaisissa peleissä on usein mahdollista omistaa jotain tavaraa, kuten miekkoja, kilpiä, haarniskoita, taloja, tontteja, autoja, vaatteita, ja niin edelleen.

Peleissä on ollut toki mahdollista omistaa *tavaraa* tai *hyödykkeitä* (goods) jo vuosikymmeniä mutta se, että pelaaminen tapahtuu satojen tuhansien jopa miljoonien muiden ihmisten kanssa tuo mukanaan suuren määrän kysymyksiä, epävarmuustekijöitä ja ongelmia.

Tässä tutkielmassa on tarkoituksena käsitellä virtuaalitavaroiden omistamisen ongelmia, niiden ostamisen ja myymisen verottamista, tällaisen virtuaaliveron oikeutusta ja oikeudenmukaisuutta sekä sen mahdollisia konkreettisia toteutuksia. Tutkielma on kirjoitettu siten, että lukijan on helppo ymmärtää ongelmien taustalla vaikuttavat osatekijät.

Tutkielman toinen luku käsittelee omistamisen ongelmaa hieman filosofisesta näkökulmasta. Pohdin omistamista ylipäätään ihmisen luomana käsitteenä ja sen yleisiä ongelmakohtia. Tällä tavalla rakennetaan raamit seuraaville luvuille, joissa tutkitaan ongelmia, joiden jokaisen pohjalla on vaikuttavana osatekijänä omistaminen. Kun pohjaongelma ja -kysymykset on asetettu, siirrytään todellisen ja virtuaalisen omistamisen vertailuun. Kolmas luku tutkii näiden kahden eroja käsitteinä ja ilmiöinä. Neljännessä luvussa pohditaan todellisten tavaroiden virtuaalista verotusta.

Viidennessä luvussa tutkitaan virtuaalista taloutta, joka on rakentunut pelien ympärille, ennen kuin siirrytään kuudenteen lukuun tutkimaan varsinaista virtuaaliverotusta, sellaisen verotusjärjestelmän konkreettisia toteutusmahdollisuuksia, vaikeuksia, hyötyjä ja haittoja. Viides ja kuudes luku menevät osittain asiasisällöiltään päällekkäin kuten myös neljäs ja viides, mutta teksti on pyritty pitämään luettavana, vaikka aihe ei olisikaan erityisen tuttu.

Viimeisessä luvussa kootaan lyhyesti yhteen kaikissa luvuissa käsitellyt asiat, saavutetut tulokset sekä pohditaan, kuinka tutkielman olisi voinut toteuttaa laajemmin sekä minkälaiset jatkotutkimukset aiheesta olisivat mahdollisia.

2. Omistamisen käsite

Omistaminen on siinä mielessä ainutlaatuinen piirre ihmisessä, ettei mikään muu eliö maapallolla koe varsinaisesti omistavansa mitään. Ihmiset ovat oravannahka-ajoista lähtien halunneet merkitä jotenkin asioita, jotka kuuluvat vain heille itselleen. Mikäli toinen ihminen ottaa luvatta toisen omistamaa tavaraa, kutsutaan sitä varastamiseksi. Jos vastaavasti esittää omanaan jonkun toisen ihmisen kirjoittamaa tekstiä, säveltämää musiikkia tai maalaamaa teosta, kutsutaan sitä plagioinniksi. Kumpikaan edellä mainituista ei olisi mahdollista, jollei ihmisellä olisi pakottavaa halua eritellä omia tavaroitaan muiden tavaroista. Jälkimmäisessä esimerkissä kyseessä on jonkin *teoksen* omistaminen. Tämä erityishuomio siksi, että on huomattavan suuri ero omistaa ainutlaatuinen ”kerran tehty” teos, kuin toistettavissa oleva liukuhihnatuote. Tällaisen teoksen omistamista on paljon helpompi tietenkin perustella kuin jonkin yhteisesti maksetun auton tai asunnon. Mutta omistaminen on jo itsessään hyvin vaikea, monisäikeinen ja väljä käsite, kuten hetken pohdinnan jälkeen selviää.

2.1. Omistamisen ongelma

Toisin kuin mikään muu eliö, ihminen haluaa, että hänellä on jotain omaa. Eläimillä ei ole mitään muuta omaa kuin jälkikasvu ja joillakin lajeilla myös reviirit, mutta niitäkään ei suoranaisesti omisteta, ei siinä mielessä kuin ihmiset omistavat omat tavaransa. Ihmiset omistavat taloja, autoja, tietokoneita, asuntoja ja vaikka mitä muuta. Tällainen asenne tavaroita kohtaan saattaa johtaa

joissain tapauksissa ristiriitoihin ihmissuhteissa, kun esimerkiksi aviopari eroaa ja aletaan keskustella mikä kuuluu kenellekin. Omistaminen on ollut jo reaalimaailman tavaroiden kanssa hankalaa, mutta koko asia vain monimutkaistuu, kun aletaan puhua virtuaalitavaroista.

Usein ihmiset käsittävät omistamisen rahan – joka on myös ihmiselle ainutlaatuinen käsite – kautta. Toisin sanoin: se, joka on jostain maksanut omistaa sen. On hyvä jo tässä vaiheessa huomauttaa, että virtuaalitavarasta ei välttämättä ole maksettu mitään, vaan se saattaa olla sattumalta löydetty. Henkilöllä on myös velvollisuus maksaa veroja omistamistaan kiinteistöstä ja ansaitsemastaan tulosta. Verotuksen perusteella voisikin ehkä pätevästi perustella, mikä on kenenkin omaisuutta. Mutta yleisen avioehdonkin perusteella varat ja velat ovat yhteisiä, joten ei sekään ratkaise ongelmaa, jonka ihmiset ovat itse kehittäneet.

Edellä todetun perusteella esimerkkinä käytetty kuvitteellinen eroava pariskunta joutuu siis edelleen jatkamaan omistusoikeuksistaan taistelemista. Aikaisemmin tässä tutkielmassa mainittiin luotu teos. Myöskään siihen liittyvällä logiikalla ei voida saavuttaa mitään kompromissia, koska kumpikaan ei ensinnäkään ole luonut taloa, minkä lisäksi talo on myös toisinnettavissa. Siinä mielessä se kuuluu sen rakentaneille ihmisille, toisaalta asukkaat ovat maksaneet sen, mutta edelleen he ottivat siihen pankilta lainaa. Ongelma tuntuu jatkuvan sitä pidemmälle mitä enemmän sitä ajattelee.

Omistamisen konkretisoituvin ongelma rationaalisella tasolla on sen määritelmän väljyys. Riippumatta siitä, kuka on maksanut veroja, kuka on kustantanut hankinnan (ja kenelle), kuka maksaa ylläpidon ja mistä materiaalit tulivat, ei voida mitenkään yksiselitteisesti osoittaa talon kuuluvan kenellekään. Juridisestihan tällaisessa esimerkissä on hyvin helppo ja jopa yksinkertainen ratkaisu: omistaja on se, joka asiakirjoihin on merkitty. Tämä ei kuitenkaan tarkoita, että asia ratkeaa oikeudenmukaisesti tai muutenkaan oikein.

Taiteessa omistaminen saattaa olla paljon selkeämmin määriteltävissä, jos ajatellaan, että omistaja on sama kuin luoja ja että ”luojuus” tuo mukanaan omistajuuden. Silloin voitaisiin uskottavasti argumentoida, että teoksen luoja omistaa oman työnsä, koska hänellä on siihen lain mukaan myös tekijänoikeus [Finlex, 2011]. Toisaalta tällöin voitaisiin väittää, että talonkin tai minkä tahansa muun, omistajuus määräytyy myös lainsäädännön perusteella. Mutta usein on osoitettu, että lakiin perustuvat päätökset eivät ole poikkeuksetta moraalisesti oikein. Se ei ratkaise omistamisen ongelmaa muutoin kuin arkisella ja juridisella tasolla. Edelleen suurin ongelma, omistajuuden ristiriitaisuus käsitteenä, jää ratkaisematta. Lisäksi, jos omistaminen pohjautuisi aina lakiin, täytyisi kaikkien, jotka antavat vanhoja tavaroitaan pois, tehdä jokaisesta jokin virallinen luovutustodistus. Muussa tapauksessa omistussuhteessa ei tapahtuisi muutos-

ta, koska sopimus olisi vain suullinen. Ystävien kesken se saattaa olla riittävä mutta lain edessä ei. Tämän tutkielman rajoissa on nyt tyytyminen kuitenkin tähän helpoimpaan ratkaisuun omistamisen kysymyksen osalta. Ratkaisusta on myös etua, sillä lähtökohdan ollessa se että luominen antaa myös omistusoikeuden tavaraan tai teokseen, saattaa se helpottaa omistamisen ongelmaa virtuaalitavaroiden suhteen.

2.2. Reaalimaailman ja virtuaalimaailman omistaminen

Reaalimaailman jokapäiväiset kulutustavarat ovat kaikki melko geneerisiä, jos niitä verrataan jonkun taiteilijan maalaamaan tauluun, jota on vain yksi ainutlaatuinen kappale. Kahdella eri henkilöllä voi olla saman valmistajan samanlainen pari lenkkitossuja, mutta on mahdotonta, että kumpikin omistaisi maalauksen *Huuto*, joka on maalattu vain kerran ja on siten todella ainutlaatuinen. Toki kummatkin heistä voivat omistaa kopiot samasta alkuperäisestä teoksesta mutta kumpikaan niistä ei vastaa alkuperäistä teosta arvoltaan eikä muullakaan tavalla. Tällaisia ei-ainutlaatuisia kopioita kutsutaan filosofiassa usein *instanssiksi* tai *manifestaatioksi*.

Toisin kuin digitaalisilla ja virtuaalisilla tavaroilla, kaikilla mitkä ovat reaalimaailmassa, on varmasti fyysinen manifestaatio. Manifestaatio voidaan määritellä esimerkiksi niin, että se on konkreettinen, kouriintuntuva vastine sille käsitteelle, joka on vain päidemme sisällä, siis ideallisen käsitteen fyysinen ilmentymä. Kun joku puhuu kännykästään, on sillä puheella jokin kohde, jokin fyysinen esine, johon puhe viittaa, eli kännykän ideallisen käsitteen fyysinen manifestaatio. On helppo myöntyä siihen, että tällainen fyysisesti olemassa oleva manifestaatio on ylipäänsä omistettavissa.

Toisaalta tässä esimerkissä on samaan aikaan kyse instanssista. Instanssi tarkoittaa tiettyä yksittäistä manifestaatiota. Monilla ihmisillä on nykyään kännykkä ja ne ovat kaikki kännykän käsitteen manifestaatioita, mutta jokainen yksittäinen niistä on sen käsitteen uusi, oma, erillinen instanssinsa. Eli kun yllä mainitun esimerkin henkilö puhuu puhelimestaan ja sen omistamisesta, puhuu hän siitä yksittäisestä instanssista, ei kenenkään muun omistamasta saman käsitteen *instantioituneesta manifestaatiosta*.

Voidaan helposti väittää, että aiemmin mainittu viittaus fyysiseen manifestaatioon ei toteudu, kun joku puhuu miekastaan, jonka löysi edellisenä päivänä WoW:ssa. Mutta kuten Lederman [2007] toteaa, on jokainen virtuaalimaailmassa manifestoituva tavara verrattavissa reaalimaailman manifestaatioihin siinä mielessä, että ne ovat samassa mielessä instansseja. Ne ovat käsitteiden yksittäisiä ilmentymiä, vaikka ilmenevätkin vain omassa maailmassaan. WoW-universumissa on käsite, vaikkapa *Cloak of Sorcery* (Taikuuden Viitta) [Lederman, 2007] ja siitä on olemassa useita instansseja, joita eri hahmot kantavat

mukanaan.

Asian voidaan sanoa toimivan aivan samalla tavalla reaalimaailman puolella. Jos joku omistaa kynän, omistaa hän yhtä lailla kynän käsitteen yksittäisen instanssin. Se instanssi on eri kuin jonkun muun omistama, vaikka kynä olisi ulkoisesti ja muulla tavalla identtinen. Täytyy myös huomata, että reaalimaailmassa oleva kynä on olemassa, vaikka sen omistaja ei olisikaan läsnä, ja edelleen se, että vastaava ominaisuus pätee myös WoW:n tavaroihin. Vaikka omistaja kirjautuu ulos, hänen maahan laskemansa tavarat, kuten Taikuuden Viitta, ovat silti olemassa pelin universumissa. [Lederman, 2007]

Tässä on kuitenkin se merkittävä ero, joka jo aiemmin tuotiin esille, että WoW-pelaaja omistaa jotain, jolla ei ole mitään fyysistä manifestaatiota. Hänen viittansa on olemassa hyvin rajoitetusti, rakennetussa bittiuniversumissa, mutta hänen kynänsä on olemassa reaalimaailmassa. Kynällä on manifestaatio riippumatta siitä, voivatko kaikki sen joskus aistineet juuri sillä ajan hetkellä aistia sitä. Viitalla puolestaan ei ole aistittavaa manifestaatiota viime kädessä ollenkaan, vaan ainoastaan virtuaalinen instanssi, jonka aistimiseksi täytyy käynnistää pelin asiakasohjelma ja siirtyä pelin universumiin. Siihen tarvitsee työkalun, synteettisen maailman. Kaikki siinä maailmassa olevat asiat myös tuhoutuvat, jos pelaajat lakkaavat pelaamasta ja maailma ajetaan alas, mutta jos kynää ei käytä kukaan sataan vuoteen, se tuskin dematerialisoituu.

Tämä on juuri virtuaalisten tavaroiden ontologinen ongelma: ne eivät ole fyysisesti olemassa, mutta ne voidaan kuitenkin aistia tietokoneen ruudulla ja niitä voidaan manipuloida pelin virtuaaliuniversumin rakennettujen lakien ja toimintojen rajoissa. Voi olla hankalahkoa tehdä loogisesti pätevää ja aukotonta argumenttia epäfyysisen tavaran omistamisesta.

3. Virtuaalitavara

Reaalimaailman tavaroista puhuttaessa usein törmää Platonin oppiin maailman kahtia jakautuneisuudesta. Hänen mukaansa on olemassa reaalimaailma, jossa on tavaroiden fyysiset manifestaatiot, erillisine instansseineen, sekä ideamaailma, jossa on ns. *universaalit*. Universaalit ovat fyysisten manifestaatioiden täydellisyyksiä, ideoita sinänsä. Ne ovat tuhoutumattomia suhteessa reaalimaailman häilyviin varjoihinsa. Voidaan esimerkkinä tarkastella tilannetta, jossa kaikki maailman kynät kerättäisiin suureen kasaan ja ne kaikki poltettaisiin. Kynän idea, ”kynyys” kuten voidaan sanoa, ei silti katoaisi mihinkään, ainoastaan kaikki sen reaalimaailman instanssit.

Tämä asia monimutkaistuu huomattavasti, kun puhutaan virtuaalisista tavaroista. Hyväksytäänpä Platonin näkemystä dualistisesta maailman rakenteesta tai ei, ovat virtuaalimaailman universaalit ja manifestaatiot silti samassa maailmassa, pelin maailmassa.

3.1. Virtuaalitavaran ontologia

Nyt kun on luotu raamit virtuaalisten ja todellisten tavaroiden ontologisille eroille sekä ongelmille, voidaan asiaa lähteä tutkimaan pintaa syvemmältä. Mikäli asenne virtuaalitavaroiden olemassaoloon on reaalimaailman tavaroiden olemassaolosta eriävä, eli ne eivät ole oikeasti olemassa, tulisi tämän loogisesti johtaa siihen, että niillä ei voi olla arvoakaan. Jos niillä ei olisi arvoa, ei tarvitsisi miettiä niiden myynti- ja ostotapahtumien verottamista. Mutta jälkimmäinen ei selkeästi pidä paikkaansa, kuten useasti on nähty, vaan virtuaalitavaroita myydään suurillakin rahamäärillä [Walker, 2007; Dibbel, 2007; Lederman 2007]. Ei vaikuta silti loogisesti pätevältä väittää, että tämä johtaa vastaavasti takaperin siihen, että virtuaalitavarat ovat olemassa. Seuraakin kysymys siitä, miten ne ovat olemassa, koska jollain tavalla niiden on oltava, muuten niitä ei voisi myydä ja ostaa.

Jotta voi omistaa, täytyy olla jotain jota omistaa. Vastaavasti, jotta voi myydä, pitää omistaa jotain, jolla on arvoa. Virtuaalitavaroiden osalta kumpikin kriteeri toteutuu, vähintäänkin kokemuksen ja ilmiön kautta. Ihmiset kokevat omistavansa virtuaalitavaransa yhtä lailla kuin autonsa ja asuntonsa. Jos joku vie luvatta heidän aseensa virtuaalimaailmassa, on se varastamista. He myös myyvät tavaroitaan [Lederman, 2007; Dibbel, 2007].

WoW ja Second Life eroavat toisistaan tavaroidensa osalta siinä, että WoW on tuotettu, sitä ylläpidetään *komentosarjoilla* (script), joilla myös rakennetaan pelin sisältö tavaroineen [Lederman, 2007]. Second Life puolestaan rohkaisee pelaajia luomaan itse maailmaansa tavaroita [Lederman, 2007]. Ontologisesti tavarat ovat samalla tavalla olemassa suhteessa toisiinsa, oli kanta niiden todelliseen olemassaoloon mikä hyvänsä.

Voidaan pätevästi argumentoida, että tavarat ovat olemassa, jos eivät millään muulla tavalla, niin virtuaalimaailmassa, jossa ne ovat luotukin. Tietojenkäsittelytieteellisesti tässä on kuitenkin toinenkin huomioitava asia. Tavarat ovat olemassa koodina *palvelimella* (server) [Lederman, 2007]. Se on konkreettisin löydettävissä oleva virtuaalitavaroiden olemassaolon muoto. Tästä voidaan alkaa vielä kiistelemään, että mitä se koodi sitten konkreettisesti on: bittejä tietokoneen muistissa, eli sähkövirtaa piireissä, eikä sähköä voi omistaa ja sen myymisestä ei siten voi mennä veroa. Vaikka sähköä kyllä myydään, mutta ei siitä itsestään silti makseta eikä veroteta, koska on mahdotonta säilöä sähköä lasipurkkiin ja pitää sitä hyllyssä. Sähkön käyttämisestä maksetaan, mutta ei sen omistamisesta.

Toisaalta näin argumentoidessa joudutaan samalla myöntymään siihen, että koska myös pelaajien *hahmot* (characters) ovat täysin identtisellä tavalla olemassa vain bitteinä, niin eikö virtuaalisilla hahmoilla voisi olla oikeus omistaa virtuaalista tavaraa ja myydä sitä virtuaalimaailmassa virtuaalista valuuttaa

vastaan? Hahmothan ovat omassa maailmassa täysin rinnastettavissa reaalimaailman henkilöihin, jotka omistavat reaalimaailmassa tavaraa ja myyvät sitä reaalimaailman valuuttaa vastaan. Edelleen, mikäli reaalimaailman tapaus on verotettavissa, miksi ei virtuaalimaailman vastineensa olisi? Jos asia olisi näin yksinkertainen, eivät viranomaiset olisikaan niin hukassa sen kanssa, ja tämä tutkielma olisi kirjoitettu turhaan.

3.2. Virtuaaliomistus

Edellisessä kohdassa päädyttiin siihen, että virtuaaliset tavarat ovat olemassa ainakin vähintään siinä maailmassa, jossa ne on luotu. Siinä samassa maailmassa, jossa pelaajien hahmot ovat samalla tasolla ja tavalla olemassa. Virtuaaliset hahmot siis omistavat virtuaalisesti virtuaalimaailmassa virtuaalista tavaraa, tämä kuulostaa ihan johdonmukaiselta – ja sekavalta. Virtuaalimaailmassa törmää usein tavaroihin, jotka ovat muotoa "*Dracula's Dagger*". Joten mikäli reaalimaailmassakin tavaran omistajuuden todistamiseen riittää suullinen sopimus ystävien kesken, on se rinnastettavissa siihen, että tavarassa olisi samantyyppinen nimitarra. Mutta virtuaalihahmot itsessään ovat reaalimaailman ihmisten omistamia ja sitä kautta kaikki muukin on hämärästi reaalimaailman ja virtuaalimaailman rajan yli heidän omistamaansa.

Kaikki mitä virtuaalimaailmassa on, on konkreettisimmillaan olemassa pelin palvelimilla bitteinä. Tästä päästään, vertauskuvaa taiteilijan luomasta teoksesta pohjana käyttäen, helposti siihen, että jos kerran miekat, kilvet, autot ja vaatteet eivät ole kooditasolla varsinaisesti edes pelaajien luomia, voivatko he omistaakaan niitä. Huomioon ottaen, etteivät he välttämättä ole myöskään maksaneet niistä mitään, jos ne on löydetty pelissä sotasaaliina (määritelty myöhemmin).

Edelleen, vaikka myönnyttäisiin siihen, että tietokonekoodia voi ylipäätään omistaa, vähintään tekijänoikeuden tai patenttien kautta, ei tavaroiden koodi silti edes ole paikallisesti heidän koneellaan. Mitä pelaajat tällöin voisivat omistaa, kun heidän hahmollaan on hallussa jonkin pelissä esiintyvän tavaran instanssi? Koodi ei ole heidän kirjoittamaa, he eivät ole tavaroista ehkä maksaneet, eivätkä ne ole fyysisesti olemassa. Toisaalta ihmiset omistavat myös yritysten osakkeita, eikä niilläkään ole välttämättä fyysistä manifestaatiota. Mutta yrityksellä itsellään on henkilöstö, toimitilat, tuotteet, patentit, jne. Ihmiset omistavat osakkeilla kirjaimellisesti osan fyysisestä kokonaisuudesta, mutta omistaessaan pelissä miekan he eivät omista osaa pelistä. Sen omistusoikeus säilyy kokonaisuudessaan pelin tehneellä yrityksellä. Instanssin omistamista ja sen pelkän käyttölisenssin omistamisen eroa tutkitaan tarkemmin luvussa 5.

Lederman [2007] tuo esille merkittävän eron World of Warcraftin ja Second

Lifen välillä. Siinä missä WoW on komentokielellä hallittu peliuniversumi, ei Second Life ole läheskään niin hallittu ja rajoitettu kokonaisuus omistamisoikeuksien näkökulmasta. WoW-pelaajat voivat vain saavuttaa tavaroiden instansseja, mutta eivät voi luoda uusia, koska pelin omistava yhtiö tekee sen. Second Lifessa puolestaan pelin omistava yritys päinvastoin rohkaisee pelaajia luomaan tosielämää muistuttavaa taloutta ja toimintaa, kuten tekemään vaatteita, taloja ja muotia sekä ostamaan tontteja, asuja ja asuntoja. Monet reaalielämän yritykset järjestävät mainoskampanjoita tämän pelin maailmassa ja ovat ”oikeasti” läsnä siellä.

Omistamisen osalta tässä on suuri ero. Mikäli aikaisempi esimerkki maalarista, joka perii omistusoikeuden tauluunsa suoraan omasta luomistyöstään, pitää paikkansa, tulisi näin tapahtua Second Life -pelissäkin. Jos joku pelaaja luo vaatteen, tulisi hänellä saman logiikan mukaan olla siihen omistusoikeus. Nyt olisi helppo argumentoida sen puolesta, että pelaaja siis myy omaisuuttaan, jonka on itse luonut, ja siitä tulisi mennä samalla logiikalla vero, kuin esimerkin taidemaalariltakin. Ei pidä kuitenkaan tehdä hätäisiä johtopäätöksiä.

World of Warcraftissa pelaajilla ei puolestaan ole minkäänlaisia omistusoikeuksia miekkaan sinänsä, sen universaaliin, koska *palveluehdoissa* (Terms of Service, ToS) pelaajat sitoutuvat tähän [Lederman, 2007]. He voivat saavuttaa miekasta jonkin yksittäisen, tai useammankin, instanssiin, mutta miekan koodi pysyy visusti Blizzardin palvelimilla, samoin kuin tekijän- ja omistusoikeudet miekkaan. Pelaajat eivät siis siinä mielessä omista miekkaa edes oman luomistyönsä kautta.

Osuva vertaus tästä tilanteesta reaalimaailmassa on DVD:n *The Matrix* omistaminen ja itse elokuvan tekijänoikeuksien omistaminen. Vaikka joku omistaisi elokuvan DVD:n, ei hän silti omista elokuvan tekijänoikeuksia. Hänellä on vain yksi tämän elokuvan tuotetuista instansseista, jonka hän voi toki myydä, jolloin menettää määräämisoikeutensa omaan kopioonsa, mutta tekijänoikeuksiin tämä ei vaikuta silti millään tavalla. [Lederman, 2007]

Tätä voidaan verrata WoW:ssa tapahtuvaan virtuaalitavaroiden myymiseen [Lederman, 2007]. Jos pelaaja myy tällaisen virtuaalisen tavaran instanssin, joka ei edes kooditasolla käy hänen koneella, kuulostaisi melko epäoikeudenmukaiselta verottaa häntä siitä, sillä eihän pelaaja myy mitään omaansa sinänsä. Se on vain hänen virtuaalisen hahmonsa virtuaalinen tavara, joka ei edes ole hänen tekemä, eikä sen konkreettisemmin hänen hallussaan, eikä fyysisesti edes olemassa. Hän vain näkee sen instanssin näytöllään pelimaailmassa. Tässä on myös se mahdollisuus, että pelaajat vaihtavat keskenään vain toisen omistamien, mutta heidän käyttöönsä tarjottujen, tavaroiden käyttölisenssejä (ks. luku 5). Kuitenkin hän saa siitä rahaa, joko pelin valuuttaa tai jopa reaalimaailman dollareita, joten tämä taas puhuisi verottamisen puolesta.

WoW:ssa on myös sellainen erikoisuus, joka voidaan jokseenkin tarkasti suomentaa *sotasaaliiksi* (loot). Sotasaalis on tavaroita, joita pelaajat saavat kentällä peitotuilta tekoälyvihollisilta, etenkin suuremmilta ns. *pomo-* tai *päävihollisilta* (boss monster, boss-class enemy) [Lederman, 2007].

Tällainen tavaroiden löytäminen, jota toki tapahtuu pelissä muutoinkin kuin vain taistelun seurauksena, asettaa vielä enemmän ongelmia omistamiselle ja myymisen veronalaisuudelle. Etiikan näkökulmasta voidaan kysyä: Jos tällaisesta sattumanvaraisesta löydöstä menisi vero siinä vaiheessa, kun se ansaittaisiin, olisiko se moraalisesti oikein? Tapausta voidaan verrata reaalimaailmassa tapahtuvaan sattumanvaraiseen todella arvokkaan jääkiekkokeräilykortin löytämiseen. Tämän todisteen valossa ei olisi oikeudenmukaista, että sotasaaliin löytämisestä menisi virtuaalivero, koska ei reaalimaailmassakaan mene samassa tilanteessa. Sattumalöydös ei ole tuloveron piirissä. Toki tämä ei vielä ratkaise kysymystä sen myymisestä ansaitun rahan verottamisesta, varsinkaan virtuaalivaluutan.

4. Vero ja reaalimaailman tavarat

Virtuaalivero on käsitteenä paljon vanhempi kuin netin yli toimivat massiivimoninpelit virtuaalitalouksineen. Sen mahdollisuuksia ja tarvetta on pohdittu jo ennen 2000-lukua. Muiden muassa Chuck [1998] pohtii, miten tulevaisuudessa tullaan verottamaan netin ylitse tapahtuvia myynti- ja ostotapahtumia. Netin ylitse tapahtuva huutokauppahan, esimerkkinä Amazon.com tai Huuto.net, on täysin verrattavissa fyysisesti tapahtuvaan vastineeseensa, ei siinä ole muuta eroa kuin alusta. Mikäli tällainen toiminta ei ole verotettavaa paikan päällä tapahtuneena, ei olisi mitään logiikkaa, että alustan vaihtuessa se olisikin. Pelien tavaroiden myyminen netin ylitse ei myöskään eroa suuresti huutokauppamaisesta kaupankäynnistä.

Samalla tavalla voidaan argumentoida, että tavaroiden ostaminen netistä tulee olla arvonlisäveron alaista, joka nykyään pitää paikkansa. Nykyään jopa ns. kasettimaksu sisältyy CD- ja DVD-levyjen, USB-muistitikkujen ja -kiintolevyjen sekä MP3-soittimien hintoihin. Lisäksi sitä ollaan tällä hetkellä yrittämässä sisällyttää myös puhelimiin. Tämä ei sinänsä ole verotusta, koska tulot tästä eivät mene valtiolle vaan Teostolle, mutta asia itsessään ansaitsee huomioinnin tässä yhteydessä.

Verotus reaalimaailman ansio- ja pääomatuloista on ainakin Suomessa jokseenkin selkeää, samoin myyntivoittovero ja arvonlisävero. Lyhyesti: kun ansaitset rahaa, verottaa valtio siitä osan, jolla pyöritetään sitten sosiaaliturvaa ja opintotukea ynnä muita tukia ja etuuksia. Myynnistä menee veroa tehdyn voiton perusteella ja arvonlisävero on sidottuna lähes kaikkiin tuotteisiin ja palveluihin.

Tämä on myös aikaisemmin esille tuodun omistuksen ongelman suhteen hyvin mutkatonta, koska ihmiset maksavat siitä, mistä tulee ostotapahtuman seurauksena heidän omaisuutta tai josta he myyntitapahtuman seurauksena luopuvat. Puhutaan siis käsitteiden fyysisistä instansseista.

Asia kuitenkin mutkistuu jo pelkästään siinä vaiheessa, kun ajatellaan, että ostetaan tavaraa netistä. Chuck [1998] tuo esille Yhdysvalloissa esiintyneitä ongelmia tässä asiassa, kuten sen, miten joissain osavaltioissa verotetaan siitä, että tavara ostetaan netistä, toisissa taas itse tuotteesta. Jos jossain osavaltiossa on voimassa kumpikin vero, maksetaan veroa viime kädessä kahteen kertaan. Tämä ei Suomen osalta ole kovin ajankohtainen asia, ainakaan enää, sillä asiat ovat – verotuksen osalta – selkeytyneet huomattavasti yli vuosikymmenessä.

Aihe on kuitenkin ilmiönä huomionarvoinen, koska yritykset saattavat säästää netin kautta tuotteitaan myydessään mm. paketointi-, painatus- tai lähettämiskuluja (kun tavaroita ei tarvitse toimittaa vähittäismyyntipisteisiin) [Chuck, 1998]. Tämä tulee huomioida yritysverotuksessa, mikäli verotuksen on määrä olla tasapuolista yrityksille ja yksittäisille henkilöille. Samanlaista se ei tietenkään voi olla, mutta sen tulee olla suhteessa yhtä suurta, progressiivista ja tilanteeseen sopivaa.

Oman ongelmansa tähän kaikkeen aiheuttaa vielä se, että kun netistä ostetaan tuotteita digitaalilatauksina, niin miten tällainen verotetaan oikein kuluttajalta, kun huomioidaan esimerkiksi *hyvitysmaksut* (royalty) musiikkilevyistä. Tässä tulee myös ilmi se ongelma, että muun muassa kaikki musiikkilevyt on mahdollista digitoida, jolloin niiden fyysinen instanssi katoaa ja tilalle tulee digitaalinen. Sitä on paljon helpompi jakaa ja levittää ynnä muuta, jolloin oikeudenmukaisen verotuksen taakka kasvaa entisestään. [Chuck, 1998]

Sama koskee pelejä. Mikäli kuluttaja lataa Steam-palvelusta pelin, kuka valvoo ja miten, että hän maksaa siitä veron, kun palvelin on toisessa maassa? Ja kumman maan veron hän maksaa: sen josta lataa vai sen jossa itse asuu? Valvonta aiheuttaa suurimman ongelman, sillä se vaatisi suuren määrän resursseja. Ja vaikka kaikki muut kysymykset voitaisiin ratkaista päädytään jälleen kysymykseen lataamisen verottamisen oikeudenmukaisuudesta kuluttajaa kohtaan. Siinä missä kuluttajaa verotettaisiin kuten aikaisemminkin, pääsisi yhtiö vähemmällä, koska se säästäisi omissa kuluissaan. Näin päädytään epätasapainoon suhteessa aikaisempaan tilanteeseen, kun kuluttajahinnat eivät laskisi, vaikka yritys pääseekin vähemmällä. Sen ei nimittäin tarvitsisi kustantaa fyysisten manifestaatioiden tuottamista enää, mutta hintojen pysyessä ennallaan kuluttaja maksaisi siitä silti.

Tässä tutkielmassa ei kuitenkaan ole tarpeeksi aikaa eikä tilaa lähteä pohtimaan näin suuria ekonomisia aspekteja, vaan keskitytään siihen, miten tämä sama ongelma ilmenee virtuaalitavaroiden myynnissä ja ostamisessa massiivi-

monipelattavien pelien universumeissa. Sillä vaikka yllä kuvailtu verotusongelma itsessään voitaisiin ratkaista, ei se vastaisi kysymykseen fyysisesti olemattoman, sattumalta löydetyn tavaran, jota henkilö ei konkreettisesti omista, myymisen tai ostamisen eikä vaihtamisenkaan verotuksesta. Sillä ei ole merkitystä sinänsä puhutaanko digitaalisessa muodossa olevasta musiikkiteoksesta vai pelin aseista tai vaatteista, sillä kummassakin tapauksessa viitataan samalla tavalla, vain sähkövirtana, olemassa oleviin tavaroihin.

5. Virtuaalitalous

Aikaisempien lukujen perusteella voidaan sanoa, että on päädytty jonkinlaiseen lopputulokseen itse omistamisen ongelmassa, virtuaalitavaroiden ontologisessa ongelmassa, sekä näiden kahden suhteesta toisiinsa. Rationaalisen ajattelun ja loogisen päättelyketjun tulos voidaan sanoa olevan, että virtuaalitavarat ovat olemassa konkreettisesti koodina palvelimella mutta käyttäjilleen ainoastaan virtuaalimaailman käsitteiden instansseina, jotka ilmenevät tietokoneen näytöllä. Käyttäjät voivat omistaa näitä instansseja (World of Warcraft) tai luoda kokonaan uusia universaaleja ja omistaa ne itse (Second Life). Riippuen tuotteen palveluehdoista, heillä on vastaavasti määräämisoikeus omistettuun instanssiin tai luotuun universaaliin.

5.1. Tapaus World of Warcraft

World of Warcraftin tapauksessa saattaa tulla mieleen kysymys siitä, miksi kukaan haluaisi ostaa mitään tavaraa, jos pelin idea on seikkailla ja löytää tai ”ansaita” tavaroita voittamalla tekoälyvihollisia. Tarvitaan perustelu sille, miksi sellaisen pelin ympärille on edes kehittynyt hyvin arvokas taloudellinen järjestelmä. Pelille ominainen edestakaisin juokseminen, tuhansien vihollisten peittoaminen ja muut hahmon kehitykselle pakolliset toistettavat toiminnat vaativat kuitenkin suuria määriä aikaa. Kyseistä toimintatapaa kutsutaan pelipiireissä ”grindaamiseksi” (grinding), joka viittaa juuri sen vaatimaan suureen määrään työtä ja aikaa sekä samojen asioiden jatkuvaan toistamiseen. Samasta syystä sellaiset pelaajat, joilla ei ole vaadittavia resursseja kulutettavana peliin, ostavat mieluummin korkealuokkaiset tavarat suoraan kuin tahkoavat peliä satoja tunteja. [Lederman, 2007]

WoW ei kuitenkaan ole ensimmäinen peli, jolla on kukoistava virtuaalitalous. Sitä edeltäneen *EverQuestin*, joka oli myös massiivimoninpeli, talouden väitettiin vuonna 2002 olleen suurempi kuin Intian tai Kiinan [Walker, 2007].

Blizzard, joka vastaa World of Warcraftin kehityksestä ja ylläpidosta, ei edes salli virallisesti kaupankäyntiä ja se onkin johtanut siihen, että pelaajat käyvät kauppaa kolmansien osapuolien kautta. WoW:n tavaroita ilmestyy myytäväksi esimerkiksi eBay-palveluun jatkuvasti. Myynti itsessään tehdään siellä, minkä

jälkeen kirjaudutaan peliin ja suoritetaan vaihtokauppa [Lederman, 2007]. Tällainen toiminta on johtanut virtuaalitalouden väistämättömään syntyyn ja sen jatkuvaan kasvamiseen, sekä arvollisesti että kokonsa puolesta.

Todellinen syy sille, miksi pelitalo ei halua tukea tavarakauppaa on se, että sellaisessa hallitussa maailmassa kuin WoW, se ei ole hyväksi pelin kehityksen suunnalle. Pelin tekijöiden on oltava koko ajan päivittämässä pelin sisältöä, jotta se pysyy mielenkiintoisena, etteivät pelaajat siirry muiden yritysten peleihin. Jos jokin ase esimerkiksi onkin odotettua tehokkaampi, täytyy WoW:n kaltaisessa maailmassa pelin *moderaattoreilla* (moderator, WoW-maailmassa myös gamemaster, GM), olla mahdollisuus muuttaa aseen tehoa tai toimintaa [Lederman, 2007]. Tämä on juuri moderaattorien tehtävä: muuttaa (*moderoida*) pelimaailmaa, jotta se säilyy mahdollisimman optimaalisena pelaamisen ja itse pelin kannalta. Tämä toteutuu, kun kooditasolla jonkin tavaran putoamis- tai löytymistodennäköisyydeksi annetaan 0.001; täten he hallitsevat pelin sisäistä ekonomiaa. Tällaista tiukkaa hallintaa ja säätelyä ei seuraavissa esimerkeissä ole. Mutta tämä on johtanut juuri nykytilanteeseen, jossa pelin epävirallinen ekonomia ulottuukin yli peliuniversumin rajojen.

Selvennykseksi on hyvä mainita, että aseiden tehokkuus pelimaailmassa paljastuu usein vasta sitten kun ammattipelaajat saavat ne käsiinsä. He osaavat *hyväksikäyttää* (exploit) pelimaailmaa ja sen lainalaisuuksia tavalla, jota pelinkehittäjät eivät tule välttämättä edes ajatelleeksi, ennen kuin näkevät jonkun toimivan siten. Jos edellisen esimerkin ase onkin tehokkaampi kuin mitä sen koodatulle harvinaisuudelle on mahdollista, johtaa se peliuniversumin taloudelliseen epätasapainoon. Nyt jos pelaajilla olisi mahdollisuus tehdä tällaisesta miekasta kopio, olisi peli tuhoon tuomittu. Vastaava esimerkki on se, miksi tavaroiden huutokauppaaminen ei ole yrityksen itsensä tukemaa, koska silloin liian heikot hahmot saavat käsiinsä liian vahvoja aseita.

5.2. Tapaus Habbo Hotel

Lisäksi on olemassa virtuaalitalouksia, jotka eivät ole varsinaisesti pelejä, vaan ainoastaan sosiaalisia universumeja, kuten suomalaisen Sulake-yhtiön *Habbo Hotel*. Habbolla on ollut vuonna 2008 päivittäin arviolta 9.5 miljoonaa vierailijaa palvelussaan, keski-iältään 15,5 vuotta [Lehdonvirta et al., 2009]. Habbon tarjoamat myyntiartikkelit tosin eroavat huomattavasti WoW-tapauksesta. Sulake myy Habbo-asiakkailleen mm. vaatteita, huonekaluja ja ”vaihtopäitä” hahmoihinsa [Lehdonvirta et al., 2009], kun taas WoW:ssa käyttäjät myyvät tavaraa keskenään. Tapaus siis eroaa verotuskeskustelun kontekstissa hieman tässä tutkielmassa aikaisemmin käsitellyistä. Blizzard haluaa todella olla hallita universumiaan ja sitä millä on arvoa. Habbon tapauksessa on helppo ratkaista verotusasia, koska on mahdollista verottaa käyttäjiä ostetusta tavarasta aivan

reaalimaailman tavaran arvonlisäveron tapaan.

Yllä mainitusta mutkattomuudesta huolimatta tapaus Habbo Hotel tuo esille yhden kiintoisimmista huomioista. Pelien pelaajat eivät ole kuluttajia sanan itse merkityksessä, sillä kuluttamiseen kuuluu sykli, jonka aikana tuote syntyy, se myydään rahaa vastaan ja sen jälkeen kulutetaan, joka johtaa sen katoamiseen [Lehdonvirta et al., 2009]. Tämä on ekonomisesti mielenkiintoista, sillä sosiaalisissa peleissä ”kulutustavaroiden” arvo ei viime kädessä laske, koska ne *eivät kulu*. Habbossa ostettu pöytä tai hame voidaan edelleen myydä pelaajalta toiselle käytön jälkeen ja sen arvo on teoriassa sama, koska se määräytyy tavaran harvinaisuuden mukaan, ei sen kunnon. Saman voidaan sanoa pätevän Second Lifen vaatteisiin tai WoW:n aseisiin ja kilpiin. Tämä on käsittämättömän suuri ero reaalimaailman tavaroiden arvon määräytymiseen. Sinänsä reaalimaailmassa on vastaavia piirteitä joillakin antiikkiesineillä. Joissain tapauksissa esineen arvo on jopa sitä suurempi mitä iäkkäämpi esine mutta siitä huolimatta ne kuluvat. Virtuaaliset tavarat ovat teoriassa ikuisia.

5.3. Tapaus Second Life

Kontrastin nimissä World of Warcraftia on hyvä verrata hallitsemattomaan peliuniversumiin, Second Lifeen. Sellaisessa hallitsemattomassa universumissa nimenomaan rohkaistaan rakentamaan omia tavaroita, luomaan taloutta ja peli itsessään tukee *linden*-valuuttansa muuntoa takaisin dollareiksi [Lederman, 2007]. Sellaisessa ympäristössä kauppatoiminnasta ei ole haittaa itse pelille. *Linden Lab*, yritys joka ylläpitää Second Lifea, tarjoaa periaatteessa vain alustan jolla toimia, mutta kaikki muu on vapaata. Tämä onkin johtanut huomattaviin investointeihin sekä reaalimaailman yhtiön mainoskampanjoihin pelin sisällä. Esimerkiksi Mazda kampanjoi autoaan pelissä ennen reaalimaailmaa [Lederman, 2007].

6. Virtuaalivero

Kysymys siitä, kuinka pelien käyttäjiä tulisi verottaa, on todellakin monisäikeinen ja siihen sisältyy paljon osatekijöitä. Olen keskustellut useammankin vapaata kilpailua kannattavan enemmän tai vähemmän talouden ammattialaisen kanssa. Heistä lähes kaikki olivat joko sillä kannalla, että kaikki tulo on verotettavaa tai ainakin verotettavissa, tai sillä kannalla, että siitä olisi vähintään hyötyä. Useimmiten kuulin perusteluna sen, ettei kenelläkään voi olla etuoikeutta myydä verottomana tavaraa, oli sillä fyysistä manifestaatiota tai ei. Monet viittasivat siihen, että tällainen tilanne täyttää yhtä lailla harmaan talouden kriteerit kuin reaalimaailmassa tavaroiden myyminen veroja maksamatta. Tähän on toki helppo myöntyä.

6.1. Käytännön toteutuksista

Ensimmäinen toteutettu virtuaaliverotusjärjestelmä otettiin käyttöön Kiinassa vuonna 2008. Veroviranomaiset päättivät, että kaikista virtuaalimaailmoissa tapahtuneista vaihtokaupoista rahaa vastaan menee 20 % veroa [Ding, 2008]. Tutkimusyhtiö iResearch Consulting Groupin mukaan virtuaalivaluuttatalous kasvaa tasaista 15–20 % vuositahtia ja tuossa taloudessa liikkuu vuosittain noin useamman miljardin edestä Kiinan yaneja [Ding, 2008]. Tämä huima kasvuvauhti on avannut bisnesmahdollisuuksia, joita on myös alettu käyttää suurella innolla.

Kiinan virtuaaliveron käyttöönotto juontaa tähän bisnesrakoon. Ihmiset ovat alkaneet palkata pelaajia pelaamaan 10-12 tunnin vuoroissa keräämään reaalimaailman rahan arvoista tavaraa WoW:n virtuaaliuniversumeista [Ding, 2008]. Tämä tavara sitten myydään, josta työnantaja kerää suurimmat tulot, mutta eivät työntekijätkään palkatta tehtäväänsä toki hoida [Ding, 2008]. On mahdollista, että tällaista toimintaa on muissakin universumeissa, mutta toistaiseksi World of Warcraft on ainoa yleisesti tiedossa oleva tapaus.

Tässä on ainoa esimerkki koko maailmassa, jossa virtuaalivero on jo käytössä. Sen käyttöönottoa on pohdittu muuallakin, etenkin Yhdysvalloissa [Walker, 2007; Dibbel, 2007], jossa pelaajia on erittäin paljon, kuten Kiinassakin. Toistaiseksi Yhdysvalloissa ei kuitenkaan ole alettu toteuttaa tällaista lainsäädäntöä.

6.2. Virtuaaliveron loogisia ongelmia

Lederman [2007] nostaa esiin useitakin ongelmakohtia virtuaaliverotuksessa, joista näkyvin on se, mitä tai mistä ihmisiä verotetaan. Erittäin olennaista olisi löytää selvä kanta sille, miten WoW:ssa pelaajien kohdalle sattuneita sotasaaliita tulisi verotuksellisesti käsitellä. Mikäli miekan instanssin löytäminen WoW:ssa ei tarkoita, että omistaa mitään konkreettista, vaan omistaa vain käyttöoikeuden siihen instanssiin (kuten mainittu luvussa 3), tulee verotuksen huomioida tämä. Muistin virkistämiseksi: samalla tavalla kuin luvussa 2 käsitellyssä esimerkissä Matrix-elokuvan DVD:n omistamisesta, ei pelaaja omista immateriaalioikeuksia miekkaan itseensä, vaan yhteen sen instantioituneeseen manifestaatioon.

6.2.1. Sotasaaliit virtuaalisina instantiaatioina

Edellä esitetyn perusteella miekka on vain virtuaalinen instantiaatio sen käsitteellisestä universaalista, täten se ei voi olla omistettavissa tavarana. Tässä tapauksessa se on tavallaan vain osoitin pelintekijöiden luomaan universaaliin. Aivan kuin ohjelmoinnissa on osoittimia tietokoneen muistissa oleviin muuttujiin tai muistipaikkoihin. Peitotun vihollisen pudottaessa sotasaalista pelaajalle

voidaan sanoa annettavan osoitin johonkin käsitteeseen ja hän manipuloi sitä, käyttäessään vain varjokopioa todellisesta. Hän ei silti omista niiden universaaleja [Lederman, 2007].

Kuulostaisi myös järkevältä, ettei sattumanvaraisia ansaintakohteita voida validisti laskea verotuksen piiriin. Jos joku löytää sadan euron setelin kadulta, onko perusteltua väittää, että hänen tulisi ilmoittaa se veroilmoituksessa? Vielä suuremman kysymyksen herättää se, onko moraalisesti oikein vaatia sellaista. Sama kysymys voidaan esittää biteistä, sähkövirran myymisestä ja verrata tätä siihen, että jollain on omasta mielestään hyvä idea ja hän myy sen sadalla tuhannella eurolla. Siis pelkkänä ideana, hän ei piirrä sitä, hän ei mitenkään luo siitä tai sen osasta fyysistä manifestaatiota, mutta silti hänen pitää maksaa veroja tuloista. Kukaan ei kuitenkaan saanut mitään, kukaan ei menettänyt mitään, mutta raha liikkui ja se on verotettavaa.

Lottovoitto ei ole verotettavaa tuloa Suomessa, joten perustelu, että sattumanvarainen ansaintatapahtuma, jolla on vielä pienempi todennäköisyys (loton seitsemän oikein todennäköisyys on yhden suhde noin viiteentoista ja puoleen miljoonaan) puolestaan olisi verotettavissa, täytyy olla melko hyvä ja aukoton. Lisäksi lottovoitto voidaan helposti käsitellä jonkin fyysisen voittamisena, vaikka Veikkaus ei annakaan seteleitä voittajalle fyysisessä muodossa. WoW:n miekka ei taas voi olla missään tilanteessa mitenkään fyysinen.

6.2.2. Sotasaaliit ansaittuna omaisuutena

Jos saaliit taas käsitellään saavutuksina, kuten maalarin taulu tai kalastajan kala, ovat ne omistettavia asioita. Mutta ei maalariakaan veroteta taulun maalaamisesta, eikä kalastajaa kalan saamisesta, eikä kuulukaan. Perustelu tälle on se, että ennen niiden arvon realisoitumista (esim. myynnin yhteydessä) ne ovat vasta investointeja. Saaliit voidaan nähdä ajan investointina peliin. Voidaan verrata tätä myös viljelijän itse kasvattamiin kasveihin, jotka hän itse kuluttaa, niitä ei myöskään veroteta häneltä. [Lederman, 2007]

Tässä tulee eteen kuitenkin ristiriita sen kanssa, mihin tutkielmassa on aikaisemmin päädytty. Maalari on tehnyt taulun, investoinut siihen, omistaa kaikki oikeudet siihen, mutta pelaaja joka on löytänyt saaliin, jonka oikeudet ovat jollain muulla, ei omista kuin instanssin, ei mitään konkreettista todellisesti, mutta pelimaailman todellisuudessa toisaalta taas kyllä. Mikäli häntä nyt verotetaan siitä, että hän myy saalistaan tai jopa siitä, että hän antaa sen pois – sillä perusteella että sillä on reaalimaailmallista arvoa – ei pelitalokaan voisi myydä, eikä luovuttaa, miekkansa koodia ilmaiseksi toiselle yritykselle, vaikka haluaisi, koska silläkin on arvoa. Voidaan arvailla kumpi näistä kuitenkin todellisuudessa olisi mahdollista tapahtua.

Toinen esimerkki: Jos Hugo Simberg eläisi vielä, maalaisi taulun ja antaisi

sen lahjaksi ystävälleen, hän ei todennäköisesti maksaisi veroa siitä, vaikka taulun hinta olisi pilvissä. Hän ei maksaisi myyntivoittoveroa taulusta, jonka antaa ilmaiseksi mutta samaan aikaan lahjoituksen kohde saattaisi joutua maksamaan tulo- tai lahjaveroa arvostetun taiteilijan arvokkaasta teoksesta. Tässä ilmenee perustavanlaatuinen ristiriita.

6.2.3. Tavaralisenssien vaihtokaupat

Mikäli tavaroiden, joita pelaajat omistavat, ajatellaan olevan vain kuukausimaksun mukana tulleita lisenssejä käyttää tavaroita, ei niiden vaihtokaupasta tulisi olla mahdollista verottaa. Siis, kun pelaaja maksaa pelin kuukausimaksun, sisältyy siihen lisensoitu oikeus käyttää peliuniversumin sisällä olevia tavaroita. Silloin pelaaja ei omistaisi *tavaroita* ollenkaan, vaan ainoastaan *käyttölisenssejä* tavaroihin. Aivan kuten työntekijöiden on mahdollista käyttää työnantajan tarjoamia tavaroita kuten pöytiä ja tuoleja. Jos pelaajat vaihtavat tällaisia käyttölisensoituja tavaroita, on se verrattavissa reaalimaailman tilanteeseen, jossa työntekijät vaihtavat työnantajayrityksensä omistamia pöytiä keskenään. Vaikka pöydät olisivat eriarvoisia, ei kumpaakaan voisi verottaa millään periaatteella uskottavasti, koska rahaa ei ole liikkunut henkilöiden välillä. He eivät ole vaihtaneet keskenään mitään omistamaansa, joten minkään tavaran arvo ei myöskään ole realisoitunut. [Lederman, 2007]

Tämä sama voidaan helposti ylettää koskemaan myös pelimaailmassa tapahtuvaa tavaroiden vaihtamista. Jos tällaisesta voitaisiin ylipäätään verottaa, olisi verotettava osuus loogisimmillaan tavaroiden arvojen erotus, tai muu ylimääräinen väliraha. Mutta tästä päästään siihen, että jos vaihtokauppa on rahaa tavaraan, ja raha on yksi peliuniversumin tavara, on se sama kuin tavara vaihdettaisiin tavaraan. Jos rahan määrä joka vaihtokaupassa liikkuu vastaa tavaran arvoa ja kummatkin ovat WoW:n sisällä olevia instantiaatioita, joihin pelaajilla on vain käyttölisenssi, on tilanne aivan identtinen. Seuraava ongelma on se, miten voidaan määrittää perusarvo jollekin tavaralle.

6.2.4. Vaihtokaupat Second Lifessa

Kun peliuniversumi vaihtuu, niin vaihtuvat ongelmatkin. Second Life ei WoW:in tavoin palveluehdoissaan pakota hyväksymään sitä, että teoriassa kaikki sen maailmassa on Linden Labin omaisuutta. WoW:n palveluehdoissa käyttäjä antaa Blizzardille teoriassa luvan käyttää jopa omalle hahmolleen keksimänsä taustatarinaa [Lederman, 2007]. Linden Lab tarjoaa myös vaihtoehdon muuttaa lindenit reaalimaailman dollareiksi. Jos joku tekee t-paidan, hänellä on täysi määräämisoikeus sen myymiseen, sen kopioinnin sallimiseen ynnä muuhun. Linden Lab ei palveluehtojen mukaan omista immateriaalioikeuksia käyttäjien luomiin universaaleihin. Mielenkiintoista on kuitenkin se, että

palveluehdoissa Linden Lab hyväksyttää kohdan, jonka mukaan pelaajat eivät omista mitään tietoa heidän palvelimiltaan [Lederman 2007]. Toisin sanoen, vaikka t-paidan tekijä omistaa immateriaalioikeudet siihen, ei hän käytännössä omista yhtäkään sen instantiaatiota, joka on koodina Linden Labin palvelimilla. Samoissa ehdoissa myös lukee, että lähettämällä sisältöä palvelimille käyttäjä antaa kaikille muille pelaajille oikeuden ja lisenssin käyttää kyseisen sisällön kopioita, vieläpä *hyvitysmaksuvapaasti* (royalty-free) [Lederman, 2007].

Jälleen törmätään kuitenkin jompaankumpaan aikaisemmin mainituista ongelmista, riippuen siitä, ovatko tavarat pelaajien omistamia vai vain käyttölisensoituja. Mikäli ensimmäinen pitää paikkansa, olisi perusteltua väittää, että t-paidan myyjä teki tuottoa myydessään vaatteen lindeneitä vastaan, koska ne ovat muunnettavissa oikeaan valuuttaan ja siihen jopa tarjotaan helppo tapa. Jos taas jälkimmäinen pitää paikkansa, ei tulo olisi siinä mielessä verotettavaa, koska se on vain jälleen kerran sama kuin aikaisemmassa työntekijöiden pöytienvaihto-operaatiossa, käyttölisenssien vaihdos tarjotun alusta sisällä, ilman omistussuhteen muutoksia.

7. Teknisistä seikoista

Kaikkien yllä kuvattujen esimerkkien pohjalla, tulisi verotuksessa itsessään olla, pystysuora (vertikaalinen) ja vaakasuora (horisontaalinen) identtisyys, mikäli tarkoituksena on noudattaa nykyisiä olemassa olevia verokäytäntöjä. Vertikaalinen identtisyys tarkoittaa lyhyesti progressiivisuutta, eli enemmän tuottoa saavia verotetaan ankarammin. Horisontaalinen identtisyys tarkoittaa puolestaan sitä, että samassa tilanteessa olevia verotetaan samalla tavalla.

7.1. Pelaajien ajallinen ja viihteellinen epätasapaino

Pelaajia on kummassakin universumissa monesta eri ikä- ja yhteiskuntaluokasta, elämäntilanteesta ja siviilisäädystä. Kaikki nämä tekijät vaikuttavat siihen, kuinka paljon pelaajilla on aikaa kuluttaa pelaamiseen, mikä on heidän investoitavissa oleva likviditeettinsä.

Sellaiset pelaajat, jotka omistavat paljon aikaansa pelille, saavat paljon enemmän tavaraa, etenkin sotasaalista, jolloin heitä verotettaisiin enemmän. Edelleen, todennäköisesti pelaamiseen on eniten aikaa opiskelijoilla, työttömillä tai osa-aikatyöläisillä, joilla on jo valmiiksi alhaiset tulot. Tällaisille virtuaaliverotus on hankala vaihtoehto. [Lederman, 2007]

Pelaajissa on kuitenkin muitakin eroja. Toiset pelaajat kuluttavat suuria määriä aikaa haaliakseen tavaraa, jotta voivat myöhemmin myydä sen reaalimaailman valuuttaa vastaan. Toiset taas pelaavat vain pelin viihteellisen arvon takia, eivätkä tuskin koskaan myy tavaraa oikeasta rahasta.

Tässä voidaan tuoda jälleen esille vertauskuva maalarista. On aivan eri asia

alkaa verottaa kotiäitiä satunnaisista maalaustensa myymisestä, tai niiden ilmaiseksi toisille lahjoittamisesta, kuin verottaa sellaista, joka tekee sitä päivittäin pääasiallisena ansaintatyönä.

Pelaajat ovat kummassakin tapauksessa suuresti epätasapainossa keskenään, mikä saattaa aiheuttaa ongelmia verotusta suunniteltaessa. Tätä ei toisaalta voida välttämättä pitää perusteluna sille, ettei virtuaaliveroa tulisi toteuttaa. Ei reaalimaailmankaan verotuksen usein väitetä onnistuneen, mutta kompromissien avulla se on kuitenkin saatu aikaiseksi ja niin toimivaksi, että siitä ei ole suurimmalle osalle haittaa.

7.3. Valuuttakurssien ja tavaroiden perusarvon sekä luonteen ongelma

Vaikka pelaajat olisivat täysin harmonisessa tasapainossa kaikella tavalla, nousee seuraavaksi ongelmaksi se, että peliuniversumien valuutta ei ole millään tavalla sidottu yhtään mihinkään, vaan sen arvo liikkuu jatkuvasti. Ei ole välttämättä edes kovin perusteltua väittää, että reaalimaailman suhteen olematon rahayksikkö tulisi sitoa kiinteästi johonkin. Lisäksi, koska siihen vaikuttaisi todella paljon pelin suosio, jos pelaajia on vähän, jos niitä on paljon, paljonko kauppaa käydään, ja niin edelleen. Mitä enemmän pelaajia, sitä enemmän ja suurempia vaihtokauppoja. Jonkun pitäisi jatkuvasti valvoa virtuaalivaluutan arvoa manuaalisesti ja laskeskella sen arvoa suhteessa reaalimaailman valuuttoihin.

Toisen ongelman valuutan arvossa muodostaa tavaroiden perusarvo. Sitä on lähes mahdotonta alkaa määritellä, koska se määräytyy suoraan pelaajien toiminnan ja sosiaalisen verkoston kautta. Ei tavaran pelkkä harvinaisuus riitä sen arvon määrittämiseen WoW:ssa. Jonkin tavaran hinta saattaa yhtäkkiä hypätä tähtitieteellisiin mittoihin, kun jokin ammattipelaaja kehittää sille hyvän hyväksikäytön kohteen tai tavan. Tällainen lisäisi entisestään koodaajien jo nyt epäinhimillisen suurta taakkaa.

Lisäksi tavarat itsessään eivät välttämättä ole mitään peliuniversumissa hahmojen kouriintuntuvia hyödykkeitä tai käsitteiden instansseja. Yritys voisi esimerkiksi myydä nettisivuja Second Lifessa ja pyytää maksuksi lindeneitä [Lederman, 2007] yrittäen kiertää veroja tällä tavalla – ja varmaan vielä onnistuen siinä. Tällaista olisi äärimmäisen hankala valvoa ja vaatimus sellaisen valvomisesta tekisi koodaajien taakasta jo yli-inhimillisen.

7.5. Tietojenkäsittelyllinen hankaluus

Jokaisessa tapauksessa, verotettiin sitten mitä tahansa ja millä tavalla tahansa, luo se joka tapauksessa suuren paineen ja taakan peliä ylläpitävälle yritykselle. Se joutuisi seuraamaan ja rekisteröimään jokaisen kerätyn tavaran ja sotasaaliin, pitämään kirjaa kaikesta ja tilittämään kaiken. Tämä ei ainoastaan asettaisi

suuria vaatimuksia jo olemassa oleville virtuaalimaailmojen ylläpitäjille, mutta johtaisi helposti uusien yrittäjien kaatumiseen.

Jos bisnesidea itsessään on tappiollinen heti alusta asti, ei ole mitään järkeä lähteä yrittämään. Liian tiukka virtuaalivero ja sen valvonta nostaisi palvelun laadun riman korkealle ja se vaatisi myös suuria alkuinvestointeja ja rahoitusta, jotta kaikki valvonta ja muu saataisiin luotettavasti toteutettua.

Lisäksi tällainen kaiken tiedon jatkuva keruu ei varmasti ilahduttaisi pelaajia, vaan heistä tuntuisi siltä, että osa heidän vapaudestaan on riistetty jatkuvan nuuskimisen seurauksena. Pelikokemus ei myöskään laitetasolla välttämättä ainakaan paranisi, kun palvelimilta vaadittaisiin äärirajoilla toimimista. Lisäksi nettiyhteyden nopeus tulisi äkkiä pullonkaulaksi uusia pelaajia houkutellessa.

7. Yhteenveto

Omistamisen ongelman ja virtuaalitavaroiden ontologisen ongelman kautta virtuaalitavaroiden verotukseen ja sen mahdollisiin toteutuksiin ja ongelmiin. Tutkielma kulki reittinsä, joka oli suunniteltukin. Aiheessa olisi runsaasti käsiteltävää, mutta aikaisemmin tehtyjä tutkimuksia on niin hämmästyttävän vähän, ainakaan saatavilla, että pitäisi melkein lähteä sitten itse tekemään sellaista. Tässä näiden parinkymmenen sivun aikana monta kertaa joutui jättämän asioita pois. Joten tutkimusaihetta kyllä riittää. Ei ollut aikaa eikä tilaa käsitellä edes viidennestä aiheen koko sisällöstä, mutta pyrkimyksenä oli nostaa esille tärkeimmät kysymykset ja ongelmat, sekä etenkin saada lukija ymmärtämään osatekijät suurimpien kysymysten takana.

Mukaan tarttui loppupäätelmä, että virtuaalivero on äärimmäisen monisäikeinen asia kaikkine osatekijöineen, eikä sen toteuttamista voi perustella vain kintaalla viitaten ja väittäen että kaikki tulo pitää verottaa. Tällainen äkkiä tekemisestä seuraava vauhtisokeus ei johda mihinkään hyvään, koskaan. Ongelma kokonaisuutena ei myöskään ole ainoastaan tietojenkäsittelyllinen, vaan siihen sisältyy paljon filosofiaa, psykologiaa (käyttäjien tuntemus ja tutkimus sekä ihmistuntemus), matematiikkaa, taloustiedettä ja pelitutkimusta.

Toisaalta voidaan johtopäätösten pohjalta väittää, etteivät virtuaalitaloudet ole vielä niin merkittävä tekijä, että niihin tarvitsisi puuttua verotuksella. Mutta toisaalta niissä on arviolta liikkunut suurempia summia rahaa kuin joidenkin valtioiden talouksissa [Chuck, 1998].

Teknisesti tällaiset ratkaisut olisivat kuitenkin todella työläitä toteutettavia, varsinkin jo olemassa oleviin peliuniversumeihin. Lisäksi se toisi sellaisia vaatimuksia laitteistotasolla, että pullonkauloja muodostuisi väistämättä pelaajakunnissa. Tähän yhteenlaskettuna virtuaalisen verotuksen vaatiman käyttäjien nuuskinnan vaikutus pelaajien tyytyväisyyteen, ei sellaisen palvelun arvosana nouse välttämättä kovin korkeaksi.

Lähteet

[Chuck, 1998] Lysbeth B. Chuck, The virtual taxman cometh: looming taxation issues for internet commerce. *Searcher* **6**, 5 (May 1998), 36.

[Dibbel, 2007] Julian Dibbel, Virtual gold could draw real taxes. *PC World* **25**, 3 (Mar. 2007), 30-33.

[Ding, 2008] Wenlei Ding, Taxing your Second Life. *Beijing Review* **51**, 48 (Nov 2008), 31.

[Finlex, 2011] Suomen tekijänoikeuslaki. http://www.finlex.fi/fi/laki/ajantasa/1961/19610404/. Tarkistettu 20.12.2011

[Lederman, 2007] Leandra Lederman, "Stranger than fiction": taxing virtual worlds. *NYU Law Review* **86**, 6 (Dec. 2007), 1620-1672.

[Lehdonvirta *et al., 2009*] Vili Lehdonvirta, Terhi-Anna Wilska and Mikael Johnson, Virtual Consumerism, *Information, Communication & Society* **12**, 7 (Oct. 2009), 1059-1079.

[Walker, 2007] Jesse Walker, Sim pickings. *Reason* **38**, 9 (Feb. 2007), 10-11.

SQL:n opetus- ja oppimisjärjestelmien arviointi ja luokittelu

Jere Myyryläinen

Tiivistelmä.
SQL:n on kehitetty helpottamaan tietokantojen käsittelyä. Se ei kuitenkaan ole tarpeeksi käyttäjäystävällinen palvelemaan kasvavaa tietokantojen hallintatarvetta. SQL:n opettamista ja oppimista varten on kehitetty useita erilaisia järjestelmiä. Tässä tutkimuksessa tarkastelen näitä järjestelmiä visuaalisuuden, automatisoinnin ja palautteen annon näkökulmista. Tarkastelun tuloksena järjestelmät voidaan jakaa täysivaltaisiin opetusjärjestelmiin ja visuaalisiin työkaluihin. Luokittelu helpottaa ohjelmistojen valintaa eri käyttötarkoituksiin.

Avainsanat ja -sanonnat: SQL, opetusjärjestelmät, oppiminen, tietokannat, luokittelu
CR-luokat: D.2.4, D.2.6, H.1.2, H.2.3

1. Johdanto

SQL-kyselyt (*Structured Query Language*) on alun perin kehitetty mahdollistamaan kyselyiden teon tietokantoihin ilman perinteisten ohjelmointikielten opiskelua. SQL:n avulla voidaan lisätä, poistaa ja muokata tietoa tietokannassa tai hakea tietokannan tietoja erilaisilla lähes selkokielisillä kyselyillä. Vaikka SQL on edelleen ohjelmointikieliä yksinkertaisempi oppia, nykymittapuulla se ei kuitenkaan ole tarpeeksi käyttäjäystävällinen palvelemaan yhä kasvavaa tietokantojen hallintatarvetta [Myers and Douglas, 2007]. Tätä tilannetta paikkaamaan kehitetään jatkuvasti uusia tapoja opettaa ja oppia SQL:n käyttöä. SQL:n oppimisen ongelmia ovat esimerkiksi käytännön oppimisen vaikeus [Hao et al., 2009a], semanttiset virheet [Myers and Douglas, 2007; Brass and Goldberg, 2005] ja tietokannanhallintajärjestelmien (*Database management system*, DBMS) puutteet [Brass and Goldberg, 2005]. Tietokannanhallintajärjestelmät ovat ohjelmia, jotka tulkkaavat käyttäjältä tulevat kyselyt tietokoneen ymmärtämään muotoon ja tämän jälkeen muodostavat kyselyä vastaavan tulosjoukon tietokannasta. Ne myös huolehtivat tietokantojen hallinnasta ilman käyttäjältä vaadittavia toimenpiteitä.

Syntaksi- eli kielioppivirheet aiheuttavat virheilmoituksen tietokannanhallintajärjestelmässä, joka ei pysty suorittamaan kyselyä. Virheilmoitus ei kuitenkaan yleensä ole kovin informatiivinen eikä sen sisältö välttämättä kerro virheen aiheuttajaa [Brass and Goldberg, 2005]. Semanttisilla virheillä tarkoitetaan syntaksiltaan oikeita kyselyitä, jotka palauttavat ei-toivotun tulosjoukon. Kyse-

ly saattaa palauttaa kaikissa tietokannan tiloissa tyhjän joukon tai muuten virheellisen tuloksen, vaikka onkin syntaksiltaan oikein. Varsinkin näitä käytännön oppimisen ja semantiikan ymmärtämisen ongelmia voidaan selittää ajan ja harjoituksen puutteella [Hao et al., 2009a].

Kyseisten ongelmien ratkaisuun on kehitelty useita toisistaan eroavia järjestelmiä. [Hao et al., 2009a; Kenny and Pahl, 2005; Abelló et al., 2008] Koska näitä järjestelmiä on olemassa jo useita, ne voidaan luokitella ja täten helpottaa järjestelmän valintaa. Tutkielmassani tutkin näitä järjestelmiä ja pyrin luokittelemaan niitä älykkyyden, visuaalisuuden, järjestelmän antaman palautteen ja sen opiskelijalle tarjoaman tuen mukaan. Tutkielman perusteella on mahdollista selvittää, millaista järjestelmää kannattaa käyttää missäkin tilanteessa.

Luvussa 2 esittelen erilaisia opetusjärjestelmiä ja tarkastelen niitä luokittelun näkökulmasta. Luvussa 3 muodostan luokittelun järjestelmien ominaisuuksien perusteella ja pyrin selvittämään, millaisiin tarkoituksiin kyseisiä järjestelmiä voisi käyttää. Lopuksi teen yhteenvedon aiheesta ja ehdotuksen aihetta käsittelevät tutkimuksen jatkamiseen.

2. Erilaisia opetus- ja oppimisjärjestelmiä

2.1. SQL-Tutor

SQL-Tutor on vuonna 1997 Canterburyn yliopistossa kehitetty järjestelmä, jonka tarkoituksena on ollut tukea opiskelijoita SQL:n oppimisessa tarjoamalla opiskelijakohtaisesti räätälöityjä tehtäviä. Se on tarkoitettu käytettäväksi perinteisen luento-opetuksen tukena, eikä sitä suositella käytettäväksi ilman teoriataustaa tietokannoista. Se tukee pelkästään SELECT-lauseita, joita käytetään, kun tietokannasta halutaan hakea tietoja. [Mitrovic, 1998]

SQL-Tutor käyttää opetuksen tukena kuutta erilaista palautetapaa. Ne muodostavat eräänlaisen portaittaisen palautteensaantikanavan, jossa opiskelija voi itse määritellä, kuinka paljon apua hän kyseiseen tehtävään tarvitsee. Aluksi SQL-Tutor antaa tehtävän suorituksesta palautteen, joka on joko negatiivinen tai positiivinen. Positiivinen palaute tarkoittaa yksinkertaisesti sitä, että tehtävä on suoritettu. Negatiivinen palaute taas kertoo, ettei tehtävä onnistunut ja ratkaisussa olleiden virheiden määrän. Seuraavalla tasolla yksi virheistä osoitetaan tiettyyn lauseen osaan, jolloin sen löytäminen helpottuu. Tämän jälkeen käyttöön otetaan vihjetoiminto. Vihjeen avulla opiskelija saa tarkempaa tietoa kyseisestä virheestä. Järjestelmä siis ilmoittaa, minkälaisesta virheestä on kyse yleisellä tasolla. Jos opiskelija ei vieläkään pysty tunnistamaan virheitä, palauttaa järjestelmä vihjeen kaikista lauseessa olleista virheistä. Lopulta oikea vastaus paljastetaan osittain ja viimeisenä mahdollisuutena täydellisesti. [Mitrovic, 2000]

SQL-Tutor on tutkimistani järjestelmistä selvästi vanhin ja täten siitä löytyy paljon tutkimusaineistoa. Sen ikä kuitenkin näkyy järjestelmän toteutuksessa. Koska SQL-Tutor on kehitetty ennen suurinta Internet-buumia, ei siinä ollut aluksi web-käyttöliittymää vaan käyttö tapahtui suoritettavalla Windows- tai Solaris-ohjelmalla. Web-käyttöliittymä liitettiin osaksi järjestelmää vuonna 2000. Mitrovicin [1998] mukaan opetusjärjestelmien tulee olla helppokäyttöisiä, vakaita ja helposti muokattavissa opiskelijan tarpeiden mukaan. Hän huomioi myös sen, että opetusjärjestelmät on tarkoitettu ongelmien ratkaisuun, jolloin niiden käyttöliittymien tulee kuvastaa ongelmien todellisia ympäristöjä. SQL-Tutorin eri versioiden käyttöliittymät ovat lähes samanlaisia ja ne on jaettu kolmeen osaan. Käyttöliittymän yläosassa järjestelmä näyttää ratkaistavana olevan tehtävän, josta opiskelija voi tarkistaa nopeasti, mitä ratkaisuun tarvitaan. Sen alta löytyvät erilliset kentät kaikille SELECT-lauseen osille, kuten SELECT, FROM ja WHERE. Samasta osasta löytyvät myös painikkeet tehtävän palautukseen, lomakkeen tyhjentämiseen ja palautteen pyytämiseen järjestelmältä. Alin osa näyttää käytettävissä olevan tietokannan taulut ja niistä löytyvät kentät. Mitrovicin [1998] mukaan tietokannan visualisointi on erittäin tärkeää, koska kaikille tietokantojen käyttäjille on varmasti tullut selväksi, että tietokannan rakenne täytyy pitää koko ajan kirkkaana mielessä. [Mitrovic, 1998]

SQL-Tutorin etu on sen opiskelija-mallintaja. Mitrovic [1998] mainitsee, että opiskelijan tietotason mallintaminen on erittäin vaikeaa, ellei mahdotonta. Kuitenkin sellainenkin mallintaminen, joka ei ole täysin tarkka tai täydellinen, voi auttaa ohjaamaan opiskelijaa eteenpäin. SQL-Tutor käyttää rajoiteperustaista mallinnusta (*Constraint-Based Modelling*, CBM), jossa mallilla pyritään mallintamaan vain opiskelijan kokemia ongelmia aiheen kanssa. Käytännössä järjestelmä pitää kirjaa siitä, millä tasolla opiskelija pystyy suorittamaan erityyppisiä ongelmia. CBM:n käyttö mahdollistaa lähes minkä tahansa loogisen ongelman tarkastamisen ja tekee täten siitä erittäin monikäyttöisen työkalun. [Mitrovic, 1998]

Mallinnusjärjestelmä vertaa opiskelijan palauttamaa tehtävää järjestelmään tallennettuun ideaalivastaukseen. Jokaiseen SQL-Tutoriin tallennettuun tehtävään on määritelty siihen liittyvät rajoitteet ja niiden tulostaulut. Tehtävän ratkaisu merkitään oikeaksi, mikäli se täyttää näiden rajoitteiden ehdot ja rajoitteisiin liittyvien tulostaulujen ehdot. Mikäli jokin rajoitteista ei täyty, tallentaa mallinnusjärjestelmä tiedon siitä opiskelijan tietoihin, jolloin malli pysyy ajantasaisena. Koko järjestelmää kuitenkin pyörittää pedagogiikkamoduuli, joka tulkitsee opiskelijamallia ja valitsee sen perusteella parhaiten opiskelijan taitoja kehittäviä tehtäviä. Pedagogiikkamoduuli myös tulkitsee opiskelijan vastaukset ja tarjoaa tehtäviä niistä ongelmista, joita opiskelija kohtaa useimmin. Opiskelijat voivat myös valita itse haluamansa ongelmatyypit ja järjestelmä etsii niihin ja opiskelijamalliin parhaiten sopivia tehtäviä. [Mitrovic, 1998]

SQL-Tutor on siis opetusjärjestelmä, jonka hyöty kasvaa sen käytön edetessä. Sen avulla voidaan opettaa SQL:n hakukyselyitä, mutta käyttäjiltä oletetaan teorialuentoihin osallistumista. Kuusiportaisen palautejärjestelmänsä ansiosta se pystyy auttamaan opiskelijaa eteenpäin mahdollisimman hienovaraisesti. [Mitrovic, 1998]

2.2. SQLTutor

Hiukan hämäävästi nimetty SQLTutor on kiinalaisen Xinzhoun yliopiston tutkijaryhmän kehittämä kokeellinen SQL-opetusjärjestelmä. Haon ryhmän mukaan opiskelijat tuntuvat aina tietävän asiat teoriassa, mutta käytännön tilanteissa teoriaa ei saada käytettyä hyväksi. Tähän tutkijaryhmä tarjoaa syyksi tietokoneopetuksessa tarvittavien tilojen ja ajan asettamia rajoituksia. SQLTutor onkin kehitetty tarjoamaan mahdollisuus käyttää virtuaalista opettajaa myös sellaisina aikoina, kun perinteinen opetus ei ole mahdollista. [Hao et al., 2009a]

SQLTutorin avulla opiskelija syöttää SQL-kyselynsä Internet-selaimeen, joka lähettää ne järjestelmän palvelimelle. Opiskelijoiden ei myöskään tarvitse asentaa tai hallita itse tarvittavia tietokantaohjelmia, vaan kaikki käyttö onnistuu Internet-selaimen kautta. Samaan aikaan opettaja pystyy seuraamaan opiskelijoiden etenemistä. Palvelimella järjestelmä tulkitsee opiskelijan kyselyt. Samalla se lajittelee mahdolliset virheet kahteen luokkaan: syntaksivirheisiin ja logiikkavirheisiin. Syntaksivirheet palautetaan käyttäjälle ajamatta niitä SQL-palvelimelle asti. Tällä pystytään säästämään rasitusta SQL-palvelimelle. Palautteeseen lisätään myös perinteistä hallintajärjestelmää tarkempi tieto virheen sijainnista. [Hao et al., 2009a]

Loogiset ongelmat ovat opiskelijalle vaikeimpia huomata. Tästä syystä järjestelmässä on päädytty ratkaisuun, jossa oikeita vastauksia ja käyttäjän kyselystä muodostuvaa vastaustaulua vertaillaan ja vertailujen tulokset kerrotaan myös opiskelijalle. Tällöin opiskelija näkee, miten kyselyssä oleva virhe voi vaikuttaa tuloksiin. Palaute on nähtävissä myös visuaalisena. SQLTutorin opiskelijanmallinnus perustuu kahteen erilliseen tekniikkaan, jotka on pyritty yhdistämään. Aluksi opiskelijasta aletaan pitää kirjaa erillisen mallin avulla. Malliin tallennetaan tietoja opiskelijan eri kognitiivisista ominaisuuksista sekä tehtävämenestyksestä. Tämän lisäksi eri harjoitusryhmien tietoja yhdistetään niin sanotuksi ”summittaiseksi pilveksi”, kun opiskelijan opinnot ovat edenneet tarpeeksi. Tämän pilven avulla opettaja pystyy tarkkailemaan opiskelijoiden tasoeroja ja selvittämään, miten yksittäinen opiskelija suhtautuu muuhun opiskelijaryhmään. [Hao et al., 2009b]

Lähes täydelliseen nimikaimaansa nähden SQLTutor on selvästi kehittymättömämpi järjestelmä. Sen palaute on melko vaatimatonta, eikä siinä ole kovin kehittynyttä automaattista toimintaa. Se kuitenkin mahdollistaa opettajalle opiskelijoiden etenemisen seuraamisen ja pitää kirjaa opiskelijoiden menestyk-

sestä ja kognitiivisista ominaisuuksista ja tämän avulla kartoittaa opiskelijoiden tasoeroja.

2.3. Automated tutoring system

Irlantilaisen Dublin Cityn yliopiston tutkijat Kenny ja Pahl ovat myös kehittäneet verkossa käytettävää opetusjärjestelmää. Sitä ei ole nimetty, mutta käytän siitä nimeä *Automated tutoring system*, jolla siihen viittaavat myös Kenny ja Pahl [2005]. Se on kehitetty täyttämään lähes samaa tarvetta kuin SQL-Tutor, mutta tutkijoiden pyrkimyksenä on ollut parantaa opiskelijalle annettavan tuen määrää ja laatua. Automated tutoring system on tarkoitettu käytettäväksi luento-opetuksen tukena ja se pyrkii toteuttamaan verkkoympäristöön sovelletun mentori-oppipoika-mallin, jossa se ottaa mentorin roolin. [Kenny and Pahl, 2005]

Järjestelmän tarkoituksena ei ole parantaa teoreettista tietotaitoa tai tuoda opiskelijan tietoon uusia ominaisuuksia, vaan se pyrkii kehittämään jo olemassa olevia taitoja. Kenny ja Pahl ovat Mitrovicin [1998] kanssa samoilla linjoilla siitä, että opetusjärjestelmän tulee kuvastaa mahdollisimman tarkasti oikeaa tilannetta. Tästä syystä järjestelmän kyselyt tehdään oikeaan SQL-ympäristöön. Opetuksen tukena järjestelmä käyttää kognitiivisia tukia (*scaffolding*). Niiden tarkoituksena on auttaa opiskelijaa alkuun uusien asioiden kanssa. Kun opiskelija etenee kyseisen asian opiskelussa, tuet häivytetään (*fading*) pois osa kerrallaan. Kennyn ja Pahlin mukaan tuet ja häivyttäminen ovat verrattavissa opettajan ja opiskelijan vuorovaikutukseen normaalissa luokkaopetuksessa. [Kenny and Pahl, 2005]

Automated tutoring system luokittelee kyselyissä olevat virheet tuttuun kahteen luokkaan. Syntaksi- ja semantiikkavirheiden jaon jälkeen virheet luokitellaan vielä tarkempiin alakategorioihin. Järjestelmään ollaan kehittämässä monitasoista virheiden luokittelutyyliä, jonka avulla virheiden vakavuutta pystytään arvioimaan paremmin ja antamaan virheistä tarkempaa tietoa. [Kenny and Pahl, 2005]

Automated tutoring system koostuu neljästä osasta: käyttöliittymästä, tarkastusosasta, pedagogisesta osasta ja opiskelijamallista. Käyttöliittymä mahdollistaa opiskelijalle mahdollisuuden käyttää järjestelmää. Perusnäkymässään järjestelmä tarjoaa opiskelijalle tehtäviä, joista tämä voi itse valita. Tehtävien viereen on kirjoitettu pieni kuvaus tehtävän tyypistä. Valittuaan tehtävän opiskelija näkee lopullisen kysymyksen. Kysymyksen alle on varattu tila vastausta varten. Kun opiskelija on lähettänyt vastauksensa, järjestelmä tarkistaa sen käyttäen erilaisia tarkastusosan erilaisia tarkastuskeinoja. Kaikkien näiden tarkastuksien tulokset palautetaan tämän jälkeen opiskelijan nähtäville graafisessa muodossa. Kaikki palautetut tiedot, opiskelijan suoritus ja muu tehtävän aikana

muodostunut data tallennetaan opiskelijamalliin. Malli siis päivittyy aina, kun opiskelija palauttaa tehtävän. [Kenny and Pahl, 2005]

Kun kaikki tieto tehtävän suorituksesta on saatu tallennettua, pedagoginen osa pääsee analysoimaan dataa. Se muodostuu kolmesta tärkeästä toiminnosta: palautteesta, ohjauksesta ja arvioinnista. Näistä palaute jaetaan kolmeen luokkaan: virheiden merkintään, vihjeeseen ja vastauksen osittaiseen paljastamiseen. Virheiden merkintä toimii järjestelmän oletuspalautteena. Se kertoo opiskelijalle, onko tehtävä hyväksytty vai hylätty ja välittää tietokannasta tulleen virheilmoituksen. Järjestelmä kuitenkin tulkitsee virheilmoitusta ensin ja muuttaa sitä helpommin ymmärrettävään muotoon. Mikäli tehtävää ei hyväksytä, opiskelijalle aukeaa mahdollisuus käyttää muita palautemuotoja. Vihje-palaute kertoo opiskelijalle, missä kohtaa virhe lauseessa on, ja pyrkii täten ohjaamaan opiskelijaa oikeaan suuntaan kertomatta oikeaa ratkaisua suoraan. Vastauksen osittainen paljastaminen kertoo opiskelijalle osan ratkaisusta. Opiskelijalle jää kuitenkin vielä tehtäväksi selvittää, miten vastausta käytetään hyväksi. Jos opiskelija ei siltikään saa ratkaistua tehtävää, voi hän lopulta tarkastaa oikean vastauksen. [Kenny and Pahl, 2005]

Kennyn ja Pahlin [2005] mukaan muut opetusjärjestelmät keskittyvät hyvin palautteeseen, mutta opastuksessa on vielä parantamisen varaa. Tästä syystä Automated tutoring systemissä on pyritty kehittämään opastus- ja arviointipuolta. Opiskelijan yrityksien määrä ja luokitellut virheet tallennetaan ja analysoidaan, jolloin opiskelijalle pystytään antamaan henkilökohtaista palautetta. Jos opiskelijalla on esimerkiksi paljon ongelmia tietyn tehtävätyypin kanssa tai sama virhe toistuu usein tehtävästä toiseen, kertoo järjestelmä, minkälaisia ongelmia opiskelijalla on ja kuinka vakavia ne ovat. Järjestelmä myös tarjoaa ensisijaisesti näihin ongelmiin soveltuvia tehtäviä, joita opiskelijan ei kuitenkaan ole pakko suorittaa. [Kenny and Pahl, 2005]

2.4. Keiron

Keiron on italialaisen Sannion yliopiston tutkijaryhmän kehittämä ohjelmisto, jossa SQL-kyselyitä voidaan suorittaa graafisen käyttöliittymän avulla. Aversanon tutkijaryhmän [2002] mukaan kyselykielet ovat vaikeita ymmärtää, jos niiden käyttäjällä ei ole kokemusta tietokannoista. Tämä voidaan kuitenkin korjata käyttämällä SQL:ään perustuvaa Visual Query Languagea (VQL) -kieltä. VQL käyttää hyväkseen graafisia elementtejä, joiden avulla tietokannan osat voidaan esittää luontevammalla tavalla ja intuitiivisemmin kuin tekstipohjaisissa järjestelmissä. Visuaalisuus tuo kuitenkin mukanaan myös rajoitteita. Sen avulla ei esimerkiksi pystytä esittämään kovin monimutkaisia kyselyitä.

Visuaaliset järjestelmät voidaan jakaa kolmeen luokkaan: taulukoiviin, kaavioilla esitettäviin ja ikoneihin perustuviin järjestelmiin. Keiron käyttää hyväkseen jälkimmäistä. Sen kehitystyön lähtökohtana on ollut luoda järjestelmä

käyttäjille, joilla ei vielä ole kokemusta tietokannoista. Kokeneemmat käyttäjät siirtyvät käyttämään mukautuvampia tekstipohjaisia kyselykieliä. Keiron ei vaadi käyttäjältä SQL:n syntaksin tuntemista eikä käyttäjän tarvitse myöskään olla selvillä käytettävän tietokannan rakenteesta. Kyselyt luodaan käyttämällä klikkaamalla, raahaamalla ja pudottamalla (*drag & drop*) ikoneita, jotka kuvastavat tietokannan eri kohteita. Järjestelmä luo ikonien perusteella SQL-kyselyn ja lähettää sen tietokannalle. [Aversano et al., 2002]

Visual Query Language -kysely muodostuu kolmesta vaiheesta: komponenttien valinnasta (käyttäjän toimesta), haluttujen ehtojen määrittelystä ja tuloksien analysoinnista. Sen kyselynäkymä on jaettu kuuteen osaan, joista ylimmässä on listattu tietokannan taulut ja niiden sarakkeet. Sen alla on erilliset ruudut, summalle, maksimille, minimille ja muutamalle muulle yleiselle SQL-funktiolle. Kolmanteen tilaan voidaan vetää ensimmäisessä tilassa olevia elementtejä. Ne kuvastavat lopputuloksena olevan tietotaulun sarakkeita. Niiden alle voidaan vetää myös ikoneita, jolloin niitä voidaan käyttää ehtoina. Lopuksi näytöllä on vielä Group by ja Order by -valikot, joiden avulla tulostaulua voidaan järjestää. Kaikista alimpana on kenttä, johon Keiron luo koko ajan dynaamisesti SQL-lauseita, jotka vastaavat näytön sen hetkistä tilaa eli vaihe kaksi suoritetaan samaan aikaan vaiheen yksi kanssa. Tämän jälkeen muodostettu SQL-kysely lähetetään järjestelmän tietokantaan. [Aversano et al., 2002]

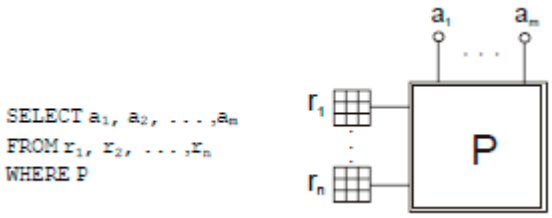
Keironin muodostaa kolme osaa: tietokantaosa, palvelinosa ja asiakasosa. Tietokantaosa muodostuu tietokannasta ja siihen liittyvästä metadatasta, jonka avulla tietokanta pystytään esittämään graafisessa muodossa. Palvelinosa toimii välittäjänä tietokannan ja asiakasosan välillä. Asiakasosa toimii käyttöliittymänä ja luo sekä lähettää tarvittavat SQL-kyselyt. Se rakentaa tietokantaosasta saaduilla tiedoilla tarvittavat graafiset elementit. Asiakasosa pystyy myös tallentamaan graafisia kyselyitä myöhempää käyttöä varten. [Aversano et al., 2002]

2.5. GraphSQL

GraphSQL on visuaalisten kyselyiden näyttämiseen kehitetty malli. Se on kehitetty italialaisessa Pavian yliopistossa. GraphSQL ei tekijöidensä mukaan ole VQL-kieli, vaan se toteuttaa SQL:n visuaalisena mallina. Sen tarkoituksena on ollut mahdollistaa SQL-kyselyiden luonti graafisia elementtejä käyttäen. Kyselyt piirretään kaksiulotteisina kaavioina, jotka esittävät kyselyt ymmärrettävässä muodossa. Tämä tukee niin uusia käyttäjiä kuin ammattilaisiakin. [Cerullo and Porta, 2007]

GraphSQL käyttää kaavioissaan erilaisia merkintöjä, jotka mahdollistavat myös vaativien kyselyiden esittämisen. Peruskyselyssä (kuva 1) käytettävät taulut merkitään ruudukolla, halutut tuloskentät pallolla ja ehdot kirjataan alueen P sisään. Monimutkaisemmissa kyselyissä käytetään myös muita merkintöjä

esimerkiksi funktioille. Sisäkkäisissä kyselyissä alkuperäistä kyselyä pyritään korostamaan kaksinkertaisella reunaviivalla. Myös useiden kyselyiden liitokset ovat mahdollisia. GraphSQL ei rajoita SQL-kieltä ja taipuu siis myös erittäin monimutkaisten kyselyiden suorittamiseen. [Cerullo and Porta, 2007]



Kuva 1. GraphSQL-kyselyn rakenne

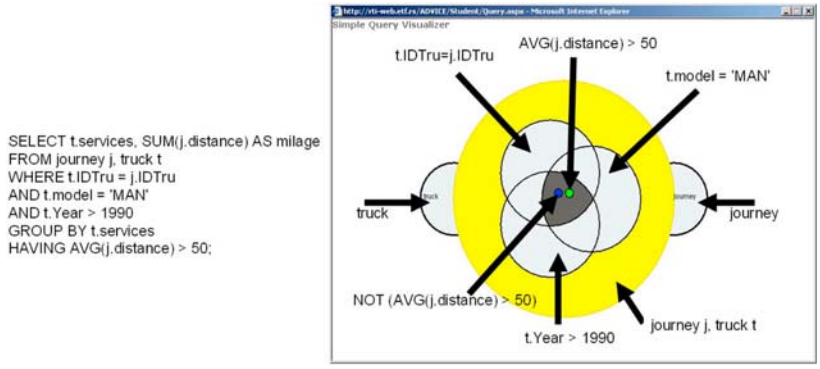
GraphSQL:n toteuttaa GraphSQL Builder, jolla graafisia kyselyitä luodaan. Sen avulla voidaan muuttaa tekstimuotoisia kyselyitä graafisiksi ja graafisia kyselyitä tekstimuotoon. Builderia ei kuitenkaan ole yhdistetty suoraan tietokantaan, joten se ei mahdollista kyselyiden suorittamista. Ulkoisesti Builder näyttää hiukan kuvankäsittelyohjelmalta. Vasemmasta reunasta löytyvät kaavioiden piirtoon tarvittavat työkalut. Niiden vieressä on piirtoalue, johon kaavioita voi piirtää. Piirtoalueen alla on tekstikenttä, johon SQL-kysely luodaan. Lisäksi oikeassa reunassa on asetuspalkki, joka kertoo valitusta elementistä tarkemmin ja mahdollistaa sen muokkaamisen. Näkymä skaalautuu erikokoisille näytöille eikä piirtoalueen kokoa ole rajoitettu. Tarvittaessa mukaan lisätään vierityspalkit. [Cerullo and Porta, 2007]

Piirretyt kuvat muuttuvat SQL-kyselyiksi järjestelmää varten kehitetyn XML-säännöstön avulla. Käyttäjän piirtämästä kuvasta muodostetaan elementtipuu, joka noudattaa tarkkaa hierarkiaa. Tästä puusta voidaan luoda sitä vastaava SQL-kysely. Tekstin muuntaminen graafisiksi elementeiksi toimii samaan tapaan, mutta siinä järjestelmän täytyy saada järjestettyä elementit järkevästi. Kaikki kuvat ovat aina täysin käyttäjän muokattavissa eikä eri elementtien sijoittelussa ole rajoitteita. GraphSQL Builder ei itsessään sisällä mitään automaattisen järjestelmän osia, kuten tehtävägeneraattoria, opiskelijan ohjausta tai tehtävien arvostelua. Se on vain työkalu, jolla SQL-kyselyistä voidaan tehdä graafisia versioita ja GraphSQL-kaavioita voidaan kääntää SQL-kielelle. [Cerullo and Porta, 2007]

2.6. ADVICE

ADVICE laajentaa opetuskohteita myös SQL:n ulkopuolelle. Se koostuu työkaluista, joilla voidaan käsitellä SQL:n lisäksi myös tietokantojen rakenteiden suunnittelua ja tietokantojen normalisointia. Opiskelijalle SQL-työkalut mahdollistavat tehtävien suorittamisen, automaattisen tarkastuksen ja kyselyiden visualisoinnin (kuva 2). Opiskelija voi myös lähettää tekemiään kyselyitä suoritettavaksi tietokannalle. Järjestelmän käyttöliittymä on melko yksinkertainen.

Ylimpänä elementtinä on valikko, josta voidaan valita, mitä ohjelman osaa halutaan käyttää. Muu käyttöliittymä muuttuu sen mukaan, mikä asetus valikosta on valittu. SQL-työkalujen kohdalla alle aukeaa kenttä, johon kyselyitä voi syöttää ja aiemmin mainitsemieni toimintojen painikkeet. Lopuksi alimpana löytyy listaus tietokannan tauluista. [Cvetanovic et al., 2011]



Kuva 2. Esimerkki ADVICE:n kyselyn visualisoinnista

Opettajalle ADVICE tarjoaa työkalut tehtävien luomiseen, opiskelijoiden seurantaan ja ryhmittelyyn sekä käyttäjien oikeuksien hallintaan. Tehtäviä voidaan luoda monien eri SQL-kyselytyyppien lisäksi myös relaatioalgebrasta, differentiaalilaskennasta, tietokantasuunnittelusta ja normalisoinnista. Opettaja myös määrittelee alkutilan, josta kaikkien opiskelijoiden suoritus alkaa. Mikäli opettaja haluaa mahdollistaa tehtävien automaattisen tarkistuksen, tulee tehtävään syöttää myös esimerkkivastaus. Esimerkkivastauksen ja alkutilamäärittelmän avulla järjestelmä tarkistaa opiskelijoiden palautuksien oikeellisuuden. Tehtävien luominen ei kuitenkaan ole pakollista ja ohjelmaa voidaan käyttää myös pelkkänä työkaluna. [Cvetanovic et al., 2011]

ADVICE toimii verkossa ja on täysin modulaarinen eli siihen pystytään lisäämään toiminnallisuutta uusilla moduuleilla. Tällä hetkellä se sisältää moduulit SQL-kielelle, muodollisille kielille, normalisoinnille, tietokantasuunnittelulle, kyselyiden visualisoinnille sekä opiskelijoiden- ja tehtävien hallinnalle. Se pystyy käyttämään tietokantanaan mitä tahansa SQL-pohjaista tietokantaa, kunhan sille on luotu tarvittavat määritykset ADVICE:n asetuksiin. [Cvetanovic et al., 2011]

2.7. LEARN-SQL

LEARN-SQL on ohjelmistoarkkitehtuuri SQL-opetusjärjestelmälle, joka tukee myös tietokannan hallintaan tarkoitettuja kyselyitä pelkän tietojen noutamisen sijaan. Toisena tavoitteena on ollut toteuttaa järjestelmä käyttäen IMS:n (*IMS Global Learning Consortium*) QTI-määritystä (*Question and Test Interoperability*), joka määrittelee interaktiivisille opetusjärjestelmille kysymyksiin ja kokeisiin tarkoitetun XML-rakenteen ja ohjelmistoarkkitehtuurin.

Järjestelmä koostuu viidestä komponentista: opettajan käytössä olevista tehtäväpankista, arviointityökaluista ja kokeidenluontityökalusta sekä opiskelijan käytössä olevista oppimisalustasta ja tehtävien palautusjärjestelmästä. Näiden lisäksi järjestelmään kuuluu vielä vastaukset automaattisesti pisteyttävä ohjelmisto. Opiskelijalle järjestelmä tarjoaa mahdollisuuden opiskella missä ja milloin tahansa, kunhan käytettävissä on Internet-yhteys. Opiskelija saa myös reaaliaikaista palautetta tehtävistä. Opettajan näkökulmasta LEARN-SQL vähentää huomattavasti tarkastukseen kuluvaa aikaa ja voimavaroja voidaan keskittää enemmän henkilökohtaiseen ohjaukseen. [Abelló et al., 2008]

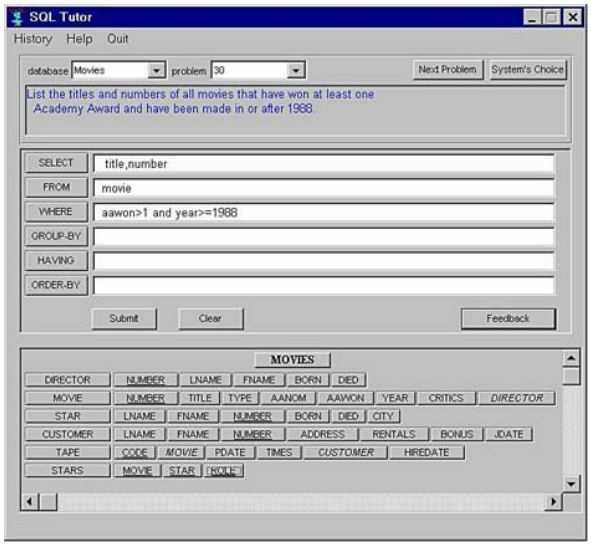
3. Järjestelmien arviointi ja luokittelu

Luvussa 2 esittelemissäni järjestelmissä on paljon samankaltaisia ominaisuuksia, joiden toteutuksessa on paljon eroavaisuuksia. Järjestelmien visuaalisuudessa on paljon eroja ja Keiron ja GraphSQL erottuvat joukosta selvästi. Myös järjestelmien älykkyydessä ja automatisoinnissa on paljon eroja, joten sitäkin on syytä tarkastella lähemmin. Suurin osa järjestelmistä antaa käyttäjälleen jonkinlaista palautetta ja ohjausta. Koska se on tärkeä osa opetusjärjestelmää, nostin myös palautteen ja ohjauksen tarkastelukohteeksi. Lopuksi jaan ohjelmat ominaisuuksiensa mukaan ryhmiin.

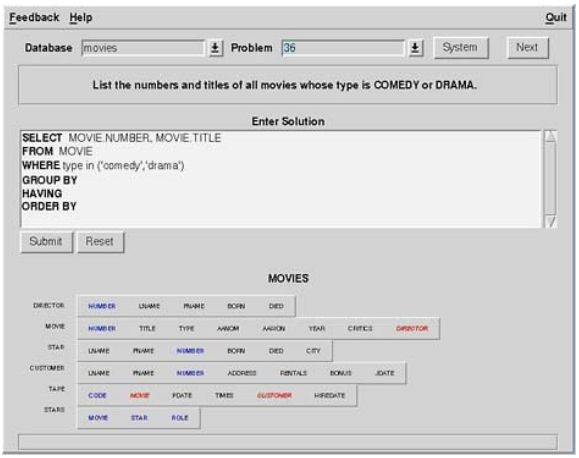
3.1. Visuaalisuus

Lähes kaikki esitellyistä järjestelmistä lupaavat visuaalista analyysiä kyselyistä tai graafista palautetta kyselyn tuloksista. Myös tietokannan graafinen esitys kuuluu suurimman osan varusteluun. Pyrin tarkastelemaan järjestelmien visuaalisuutta palautteen ja opastuksen annossa sekä muiden graafisten elementtien hyväksikäyttöä. Lisäksi tarkastelen ulkoasun selkeyttä ja sitä, kuinka käyttöliittymällä on pyritty helpottamaan oppimista.

SQL-Tutorin vanhuus näkyy sen visuaalisuudessa. Sen ulkoasu on selkeä, mutta siinä ei ole käytetty graafisuuden tuomia hyötyjä hyväksi. Tietokannan rakenne on esitetty ikkunan alalaidassa eräänlaisena taulukkona, mutta sen esitystapa eroaa käyttöjärjestelmäversioiden välillä. Windows-versiossa (kuva 3) käyttöliittymässä ei käytetä hyödyksi edes värejä, vaan tekstikohteita korostetaan alleviivauksin ja kursivoiden. Solaris-version (kuva 4) ulkoasu käyttää grafiikkaa paremmin hyväkseen. Sen tietokantanäkymä on selkeä ja käyttää värejä merkitsemään taulukoiden avaimia ja viitekenttiä. Verkkoversio on lähes identtinen Windows-version kanssa, mutta käyttää keltaista tekstiä punaisella pohjalla. Kaikissa versioissa SELECT-lause on jaettu selkeisiin osiin. Tämä vähentää syntaksivirheitä, jotka johtuvat väärässä järjestyksessä olevista lauseen osista.



Kuva 3. SQL-Tutorin Windows-version ulkoasu

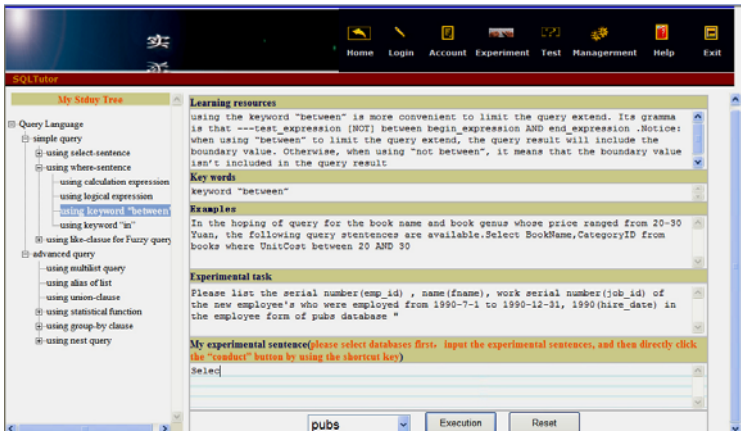


Kuva 4. SQL-Tutorin Solaris-version ulkoasu

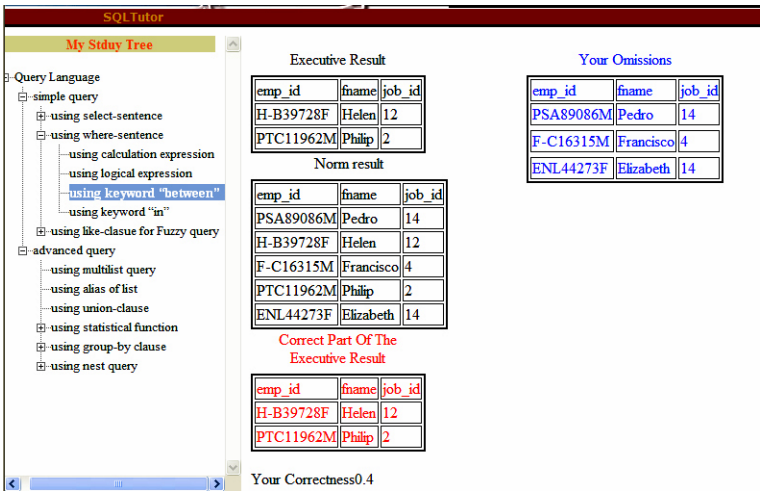
Vaikka SQL-Tutor tarjoaa monitasoisen kanavan palautteelle [Mitrovic and Martin, 2000], ei palautteen annossa käytetä graafisuutta hyväksi. Kaikki palautteet koostuvat tekstistä, jossa kerrotaan sanallisesti, missä kohtaa kyselyä virhe sijaitsee tai miten opiskelijan kannattaa seuraavaksi edetä. Voidaan siis sanoa, ettei SQL-Tutor käytä visuaalisuutta juurikaan hyväkseen.

SQLTutor on kaimaansa hieman visuaalisempi. Sen käyttöliittymässä käytetään selvästi enemmän värejä ja osa toiminnoista on merkitty myös ikonein. Perusnäkymässään (kuva 5) käyttöliittymän vasemmassa reunassa on sivupalkki, josta opiskelija näkee selkeästi, mitä tehtävää ollaan suorittamassa ja kuinka tehtävät etenevät sen jälkeen. Tehtäväikkunaan on mahdutettu tehtävänannon ja vastauskentän lisäksi myös esimerkkejä ja ohjeistusta, jotka vievät melko paljon tilaa itse tehtävän suoritukseen tarvittavilta työkaluilta. Palautteen annossa on käytetty hyväksi värejä ja taulukoita (kuva 6), joista opiskelija pystyy melko helposti tarkastamaan kyselynsä tulostaulun ja vertaamaan sitä esimerkkivastauksen tulostauluun. SQLTutor käyttää kaimaansa paremmin visuaalisuutta hyväkseen, mutta siitä puuttuu SQL-Tutorissa oleva tietokannan

visualisointi. Opiskelijan täytyy siis muistaa tietokannan rakenne ulkoa tai kirjata se itselleen ylös muistamisen helpottamiseksi.

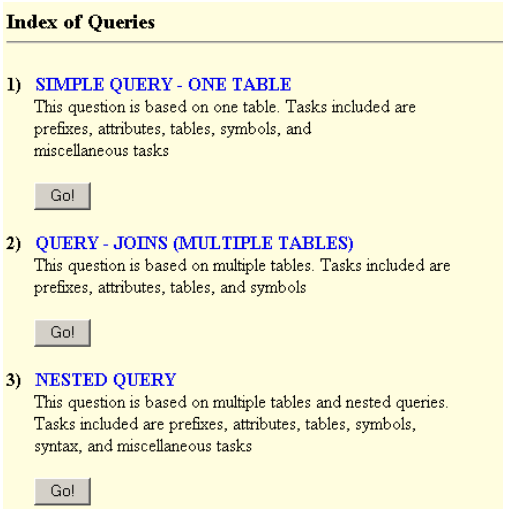


Kuva 5. SQLTutorin tehtävänäkymä

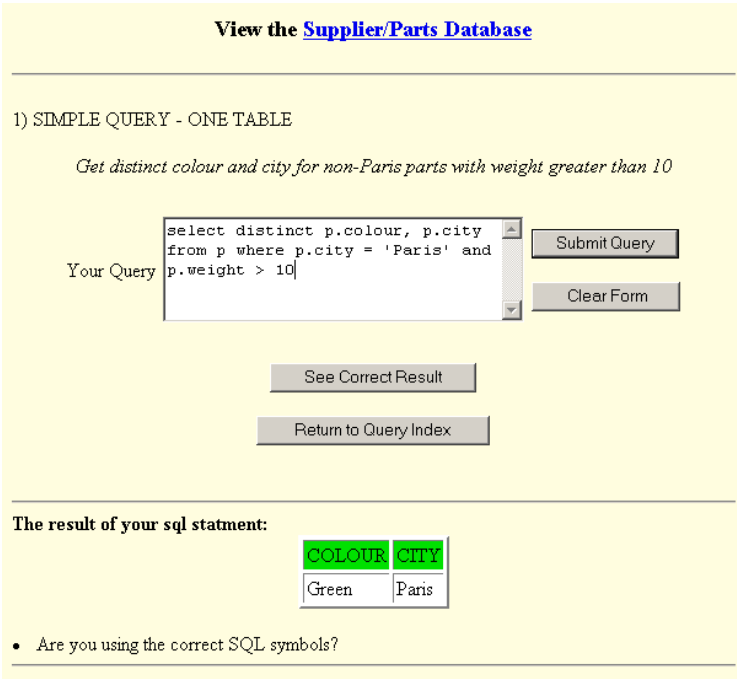


Kuva 6. SQLTutorin palautenäkymä

Automated tutoring systemin käyttöliittymä on todella yksinkertaistettu. Alkunäkymässään (kuva 7) se ei tarjoa tehtäviin siirtymisen lisäksi mitään muuta. Tehtävänäkymä (kuva 8) on myös hyvin pelkistetty. Se on jopa SQLTutoria yksinkertaisempi ja sisältää vain tehtävänannon, vastauskentän ja painikkeet mahdollisiin toimintoihin. Tehtäväsivulta on mahdollista saada näkyviin käytettävän tietokannan rakenne, mutta se vaatii erillisen linkin klikkaamista. Kaikki palautteet ja vihjeet lisätään vain sivun alalaitaan. Vaikka järjestelmä käyttääkin värejä, eivät ne edistä oppimista tai helpota hahmottamaan tärkeitä elementtejä. Väreillä vain korostetaan otsikoita ja linkkejä.



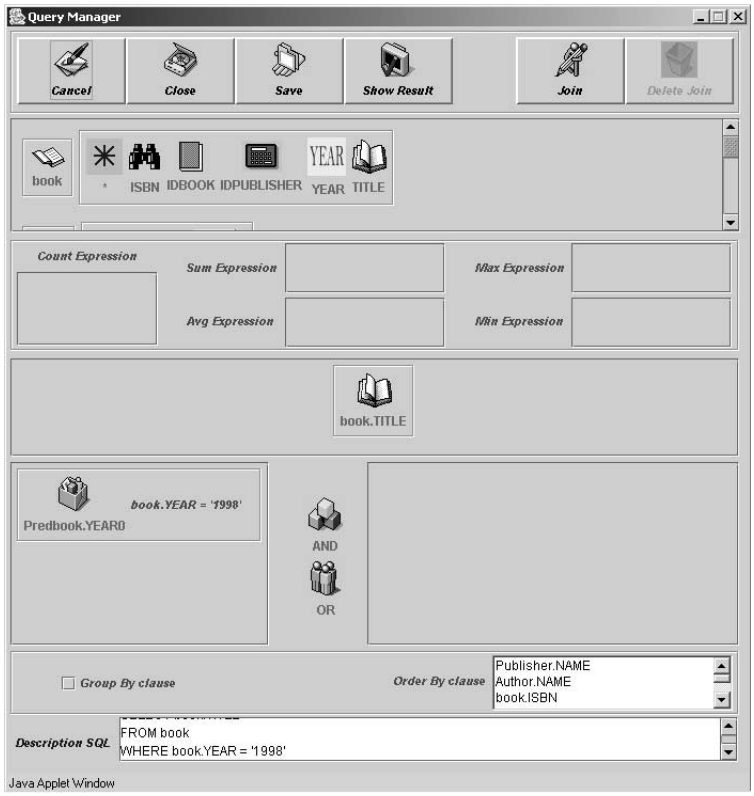
Kuva 7. Automated tutoring systemin valikko



Kuva 8. Automated tutoring system: Tehtävänäkymä, jossa on käytetty vihjetoimintoa

Keiron käyttää ikoneihin perustuvaa visuaalista esitystapaa, minkä takia se ei vastaa ulkoasultaan muita tarkastelemiani järjestelmiä. Sen käyttöliittymä (kuva 9) on selkeä ja käyttää hyväkseen erilaisia rajattuja alueita, jotka helpottavat eri osien hahmottamista. Liikuttelemalla ikoneita alueelta toiselle saadaan aikaiseksi kyselyitä. Kaikissa toimintopainikkeissa on tekstin lisäksi ikoni kertomassa painikkeen toiminnasta. Keironin visuaalinen tyyli auttaa tietokantoihin vähemmän tutustuneita aiheen alkuun, mutta se rajoittaa kyselyiden monimutkaisuutta niin paljon, ettei sitä pysty hyödyntämään edistyneempien opiskelijoiden opetuksessa. Keiron ei mahdollista tehtävien määrittelyä eikä

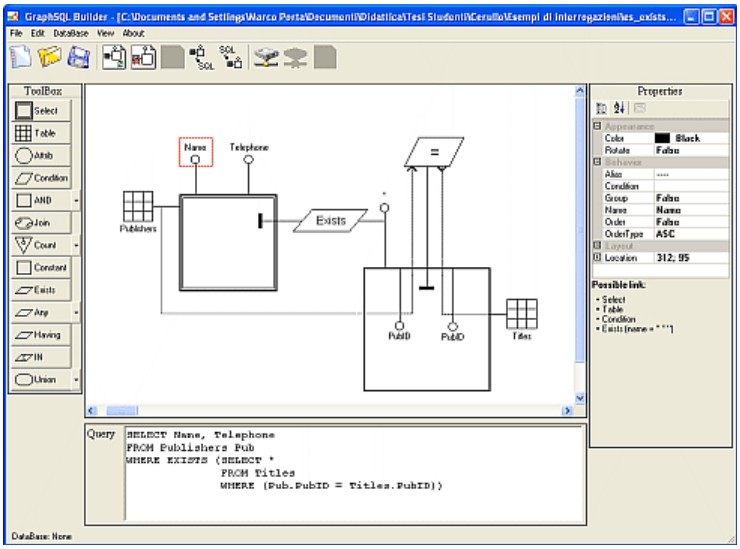
täten pysty tarkistamaan tai antamaan ohjeistusta kyselyiden tekoon, joten näiden osien visuaalisuutta ei ole mahdollista arvioida.



Kuva 9. Keironin kyselyikkuna

GraphSQL Builder on Keironin tapaan vain graafinen työkalu, jolla pystytään luomaan SQL-kyselyitä GraphSQL-notaatiota hyödyntävää graafista editoria käyttäen. Sen käyttöliittymä (kuva 10) on selkeä ja tuo mieleen kuvankäsittelyohjelmistot. Piirrettyjen elementtien kokoa ja väriä voi muuttaa, jolloin elementtejä voi määritellä oman mieltymyksen mukaan. Tämä mahdollistaa esimerkiksi tärkeiden elementtien korostamisen tai usein toistuvien osien häivyttämisen taustalle, jolloin voidaan keskittyä kyselyn tärkeimpään osaan. GraphSQL Builderilla ei ole mahdollista suorittaa kyselyitä, joten sen visuaaliset osat rajoittuvat piirtonäkymään.

ADVICE:n SQL-työkalut (kuva 11) ovat visuaalisesti läheistä sukua SQL-Tutorille, SQLTutorille ja Automated tutoring systemille. Näkyvillä on vain vastauskenttä, toimintopainikkeet ja tietokannan taulujen ja kenttien listaus. ADVICE:n visuaalisuuteen ei ole kiinnitetty huomiota kovinkaan paljoa, sillä se ei käytä hyväkseen värejä tai graafisia esitysmuotoja. Ainut visuaalisuutta hyödyntävä osa on SQL-kyselyn visualisointityökalu (kuva 2). ADVICE:sta löytyneet dokmentit eivät sisältäneet minkäänlaista kuvausta ADVICE:n antaman palautteen muodosta, joten en pysty arvioimaan sen visuaalisuutta.



Kuva 10. GraphSQL Builderin käyttöliittymä

Kuva 11. ADVICE:n SQL-työkalut

Visuaalisesti järjestelmät eivät siis paljon eroa toisistaan. Täysin graafiseen esitystapaan perustuvat Keiron ja GraphSQL toimivat selkeästi pelkkinä työkaluina, joiden avulla kyselyitä voidaan kirjoittaa graafisesti. LEARN-SQL on ohjelmistoarkkitehtuuri, eikä täten sisällä valmista käyttöliittymää. Muut järjestelmät käyttävät grafiikkaa ja visuaalisia elementtejä vähemmän hyödyksi. Niiden visuaalisuus rajoittuu yleensä lähinnä muutamaan kohteeseen, eikä visuaalisuutta päästä hyödyntämään kunnolla.

3.2. Älykkyys ja automatisointi

Suurin osa esitellyistä järjestelmistä pyrkii automatisoimaan tehtävien tarkastusta ja antamaan opettajalle mahdollisuuden keskittyä enemmän henkilökohtaiseen opetukseen. Tarkastelen automatisoinnin laajuutta ja sen takana olevien toimintojen älykkyyttä. Pyrin myös selvittämään, kuinka paljon järjestelmästä on hyötyä opettajalle.

SQL-Tutor tarkistaa automaattisesti opiskelijoiden vastaukset. Se vertaa opiskelijan tarjoamaa vastausta järjestelmään tallennettuun esimerkkivastauk-

seen. Vertailussa ei pelkästään verrata vastauksia toisiinsa, vaan opiskelijan vastaus ajetaan tehtävään liittyvien rajoitteiden läpi, jolloin automaattinen tarkastus pystyy myös ohjaamaan opiskelijaa oikeaan suuntaan. Esimerkkivastausta ei generoida, vaan opettaja joutuu itse laatimaan kysymyksiinsä myös vastaukset. Järjestelmä pitää kirjaa opiskelijan etenemisestä erillisellä opiskelijamallilla, jonka avulla opettaja pystyy seuraamaan opiskelijan etenemistä ja näkemään, millaisissa tehtävissä opiskelijalla on eniten ongelmia.

Myös SQLTutor sisältää tehtävien automaattisen tarkastamisen. Se vertaa opiskelijan kyselystä saatua tulostaulua esimerkkikyselyllä tehtyyn tauluun ja tämän jälkeen määrittelee, kuinka suuri osa esimerkkitaulusta kuului opiskelijan tulostauluun. Vertailu on SQL-Tutoriin nähden selkeästi suoraviivaisempi, eikä se ota kantaa siihen, minkä tyyppinen virhe on kyseessä ja miten opiskelijan kannattaisi edetä asiassa. Opettajan kannalta SQLTutor toimii hyvänä työkaluna opiskelijoiden taitotason tarkkailuun. Se kerää kaikista opiskelijoiden yrittämistä tehtävistä tietoa ja päivittää opiskelijamallia niillä. Järjestelmä vertaa opiskelijan etenemistä muihin saman ryhmän opiskelijoihin ja arvostelee opiskelijan taitoja eri osa-alueilla.

Automated tutoring system toimii lähes SQL-Tutorin ja SQLTutorin kaltaisesti. Se tarkistaa tehtävät ja antaa niistä palautetta automaattisesti. Tämän lisäksi se kirjaa ylös kaiken tehtävistä saadun informaation ja tallentaa sen opiskelijamalliin. Automated tutoring system eroaa SQLTutoreista kuitenkin siinä, että se myös opastaa opiskelijoita automaattisesti. Kun järjestelmä on tallentanut opiskelijamalliin tarpeeksi kattavan määrän tietoa, alkaa se ohjata opiskelijoita sellaisiin tehtäviin, joiden suorituksessa heillä voi olla eniten ongelmia. Se myös informoi opiskelijoita tekemistään huomioista, jotta opiskelija pystyisi keskittymään näiden virheiden korjaamiseen. Automated tutoring systemin opettajalle tarjoamat työkalut on jätetty kokonaan mainitsematta järjestelmää esittelevässä julkaisussa [Kenny and Pahl, 2005].

ADVICE tukee SQL:n lisäksi myös muita tietokantojen opetukseen liittyviä asioita, kuten tietokantasuunnittelua ja tietokannan normalisointia käsitteleviä tehtäviä. SQL-työkalujen avulla voidaan tarkistaa opiskelijoiden vastaukset automaattisesti. ADVICE pyrkii selvittämään, mitä eroa opiskelijan vastauksella ja sen muodostamalla tulostaululla on suhteessa esimerkkivastaukseen ja sen tulostauluun. Järjestelmä käyttää kahta erilaista tarkastustyyppiä tehtävien tarkastamiseen riippuen siitä, minkä osa-alueen tehtävä on tarkastettavana. SQL-kyselyiden tarkastus suoritetaan vertailemalla saatua tulostaulua esimerkkivastaukseen liitettyyn tulostauluun. Tämä vertailutapa on valittu, koska ADVICE mahdollistaa myös tietokannan tietojen muokkaamisen pelkän tietojen noutamisen sijaan. ADVICE pitää kirjaa kaikista opiskelijoiden vastauksista ja tarjoaa opettajalle mahdollisuuden tarkastella opiskelijoiden etenemistä koko ohjelman käytön ajan. Opettaja pystyy tällöin auttamaan sellaisia opiskelijoita, jotka vai-

kuttavat tarvitsevan lisäopastusta. Harjoitusten lopuksi opettajalle tarjotaan raportti kaikista harjoitukseen osallistuneista opiskelijoista, joka sisältää kaikki opiskelijoiden vastaukset annettuihin tehtäviin. Tarkastusvaiheessa opettaja voi käyttää järjestelmän automaattista tarkastustoimintoa.

LEARN-SQL:sta tarjolla oleva materiaali (ks. esim. [Abelló et al., 2008]) ei kerro järjestelmän automatisoinnista kuin sen, että tehtävät pisteytetään automaattisesti. Opettajalle se tarjoaa mahdollisuuden luoda testejä, lisätä tehtäviä ja valvoa opiskelijoiden etenemistä. Abelló ja muut [2008] lupaavat, että opettajan aikaa ei kulu tehtävien tarkastamiseen tai suoritusten arvioimiseen, mutta se on myös mahdollista tehdä tarvittaessa manuaalisesti.

Keiron ja GraphSQL Builder eivät sisällä automatiikkaa, jonka avulla pystyttäisiin nopeuttamaan tehtävien tarkastamista tai antamaan palautetta. Ne ovat vain työkaluja erikseen annettujen tehtävien suorittamiseen.

Kaikki automatiikkaa sisältävät järjestelmät rajoittuvat lähinnä tehtävien automaattiseen tarkastamiseen. Automated tutoring system antaa myös opiskelijoille automaattisesti palautetta oppimisen etenemisestä. ADVICE taas tarjoaa opettajalle mahdollisuuden tarkkailla opiskelijoita koko harjoituksen ajan. Myös LEARN-SQL tarjoaa alustassaan mahdollisuuden kokeiden tekemiseen ja automaattiseen pisteytykseen.

3.3. Palautteen anto ja opiskelijoiden avustaminen

Yksi opetusjärjestelmien suurimmista hyödyistä on se, että ne pystyvät opastamaan opiskelijaa ja antamaan opiskelijalle palautetta tehtävistä myös silloin, kun se ei ole opettajalle mahdollista. Palautteenantamistavat vaihtelevat eri järjestelmien välillä jonkin verran. Jotkin järjestelmistä osaavat antaa myös pitkäjänteisempää palautetta. Tarkastelen näiden kahden palautteenantamistavan lisäksi myös järjestelmiä esittelevissä julkaisuissa käsiteltyjä käyttötestauksia, jotka antavat näkökulmaa järjestelmän oikeasta hyödystä.

SQL-Tutor on kehittänyt omasta palautteenantotavastaan kuusiportaisen. Se pyrkii eri portaiden avulla tarjoamaan opiskelijalle aina mahdollisimman hienovaraisen vihjeen avustaessaan opiskelijaa eteenpäin. Opiskelija pystyy itse määrittelemään haluamansa palautetason, lähtien pelkästä oikein/väärin -palautteesta aina esimerkkivastauksen esittämiseen asti. Tällä tavalla pyritään toisaalta ohjaamaan opiskelijaa oikeaan suuntaan ja toisaalta antamaan opiskelijalle mahdollisuus myös konkreettisempaan apuun, jolloin tehtävään ei ehdi turhautua.

Mitrovic ja Brent [2000] ovat selvittäneet SQLTutorin eri palauteportaiden vaikutusta opiskelijoiden kykyyn ratkaista tehtäviä. Tutkimus on suoritettu kolmessa ryhmässä, joista yhdelle tarjottiin mahdollisuutta käyttää vain palautetasoja 1 ja 2 (oikeellisuus ja virheen kohdistus oikeaan lauseenosaan). Toinen ryhmä sai käyttöönsä palautetasot 3 ja 4 (vihje ja kaikkien virheiden esittely

portaan 2 tapaan) ja kolmas ryhmä palautetasot 5 ja 6 (osittainen- ja täysi esimerkkivastaus). Tutkimustulokset (taulukko 1) osoittavat, että ryhmä 2:n koehenkilöt suorittivat tehtävät muiden ryhmien koehenkilöitä nopeammin ja muita ryhmiä pienemmällä määrällä vastausyrityksiä. Ryhmä 3:n koehenkilöt saivat useimmin oikean palautettua oikean vastauksen. Tämän lisäksi palautetasoja vertailtiin myös toisesta näkökulmasta. Tämän tutkimuksen tarkoituksena oli selvittää, kuinka hyvin opiskelija suoriutuu samankaltaisista tehtävistä palautetason käyttämisen jälkeen. Tuloksista (taulukko 2) selvisi se, että opiskelijat, jotka olivat käyttäneet tasoja 2 - 4, pystyivät muita tasoja käyttäneitä opiskelijoita paremmin suorittamaan kesken olleen tehtävän ja muut samaa tehtävätyyppiä käyttävät tehtävät. Tason 5 tulokset olivat myös hyviä, mutta tutkijoiden mukaan niiden otanta on liian pieni, jotta siitä voitaisiin tehdä luotettavia johtopäätöksiä. Taso 6:n todettiin olevan lähes haitallinen opiskelijoiden oppimiselle. [Mitrovic and Brent, 2000]

Ryhmä	**Tehtävistä ratkaistu**	**Hylätyt yritykset**	**Yritykseen kulunut aika**
Ryhmä 1	84,10 %	2,24	78,05 s
Ryhmä 2	83,49 %	2,16	47,80 s
Ryhmä 3	87,07 %	2,21	65,26 s

Taulukko 1. SQL-Tutorin palautetasojen arviointi [Mitrovic and Brent, 2000]

Palautetaso	**virheellisesti tehtyjen tehtävätyyppien määrä**	**Hyväksytyt yritykset**	**Hylätyt yritykset**	**Selvitti tehtävän**	**Selvitti muut vastaavat tehtävät**
Oikein/Väärin	436	254	636	29 %	78 %
Virheen merkintä	116	98	126	44 %	81,8 %
Vihje	43	33	43	43 %	74,4 %
Kaikki virheet	64	72	81	47%	80 %
Osittainen vastaus	26	22	10	69 %	91,6 %
Täysi vastaus	18	6	16	27 %	44,2 %

Taulukko 2. SQL-Tutorin palautetasojen vaikutus oppimiseen [Mitrovic and Brent, 2000]

SQLTutorin palautteenanto ei ole aivan SQL-Tutorin laajuista. Se kertoo opiskelijalle vain sen, mitkä rivit opiskelijan lähettämän vastauksen tulostaulussa kuuluvat oikeaan tulostauluun ja mitkä eivät. SQLTutor kuitenkin tarjoaa ohjeistusta suoritettavana olevan tehtävätyypin ratkaisuun. Ohjeistus on nähtävillä samassa näkymässä kuin vastauksien syöttö, joten se on opiskelijan käytet-

tävissä koko ajan. Ohjeistus sisältää kuvauksen tehtävätyypistä, listaa tehtävään liittyvät avainkohdat tai -sanat sekä esimerkin, jossa avainsanoja käytetään oikeassa kontekstissa. SQLTutoria on testattu yhdellä raportoidulla käyttäjätestillä. Testissä 254 opiskelijaa jaettiin kahteen ryhmään ja heistä 158 ottivat osaa SQLTutor harjoituksiin. Loput 96 opiskelijaa ottivat osaa vastaavaan testiin, mutta käyttivät sen tekemiseen normaalia SQL-ympäristöä. Tutkimuksen mukaan SQLTutoria käyttäneet opiskelijat suorittivat testin 14,8 % paremmin kuin kontrolliryhmän opiskelijat. [Hao et al., 2009]

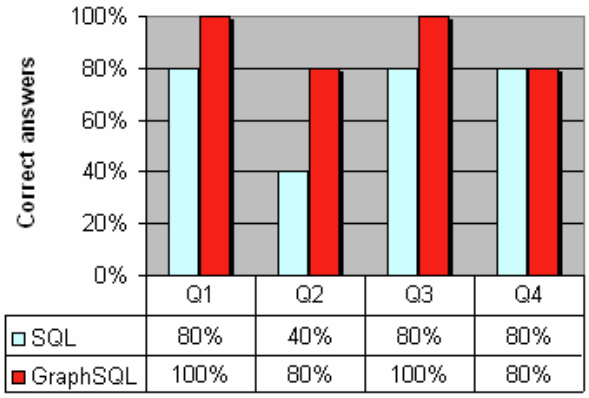
Automated tutoring systemin avustavat toiminnot ulottuvat tehtävistä saatavan palautteen lisäksi myös opiskelijan ohjaukseen. Se tarjoaa kolmiportaisen palautekanavan. Palautekanavan perusidea on sama kuin SQL-Tutorissa, eli tarjota opiskelijalle vihjeitä tehtävän ratkaisuun. Sen tasot vastaavat SQL-Tutorin palautetasoja 3, 4 ja 5. Kuten SQL-Tutoria koskevasta tutkimuksesta kävi ilmi, edistivät nämä tasot parhaiten opiskelijoiden oppimista. Palautteen lisäksi Automated tutoring system osaa myös kertoa opiskelijalle, missä osa-alueissa tällä on eniten ongelmia ja tarjoaa niihin soveltuvia tehtäviä. Tämän ansiosta opiskelija pystyy paremmin seuraamaan sitä, millaisiin ongelmiin tehtävien teossa kannattaa kiinnittää huomiota. Järjestelmä pyrkii siis olemaan opiskelijan tukena koko ajan.

Automated tutoring systemistä on tehty myös käyttäjätutkimus. Sen aineisto kerättiin haastatteluilla, järjestelmän lokitiedostoista sekä analysoimalla opiskelijoiden vastauksia. Tutkimustuloksista tehtiin muutamia huomioita. Ensinnäkin noin 20 % järjestelmän käytöstä tapahtui viikonloppuisin, jolloin perinteistä opetusta ei ole mahdollista järjestää. Käyttäjistä 84 % ilmoitti käyttäneensä järjestelmän tarjoamia kognitiivisia tukia. Käyttötapojen lisäksi myös opiskelijoiden mielipide verkko-opetukseen selvitettiin. Vastaajista 85 % piti verkko-opetusta tärkeänä osana tietokantoihin johdattavaa kurssia. Tutkimuksen osallistuneiden opiskelijoiden vastauksien perusteella ei kuitenkaan pystytä sanomaan, onko verkko-opetus parempi vaihtoehto kuin tietokoneluokassa tapahtuva lähiopetus. Opiskelijoiden opintomenestys on parantunut järjestelmän käyttöönoton myötä vuosittain 2 % neljän vuoden aikana. Tänä aikana järjestelmää on kehitetty eteenpäin, mutta kurssin muut osa-alueet ovat pysyneet muuttumattomina. [Kenny and Pahl, 2005]

Koska Keiron ei itsessään ole opetusjärjestelmä vaan pelkkä työkalu kyselyiden tekemiseen, ei se sisällä myöskään minkäänlaista palautteenantotapaa opiskelijalle. Sen käytöstä on tehty tutkimus. Tutkimukseen osallistuneet opiskelijat olivat mukana aihetta käsittelevällä kurssilla ja omasivat lähes samantasoiset lähtötaidot. Tutkimus suoritettiin tenttitilaisuutena, jossa kaikille opiskelijoille annettiin samat tehtävät. Jokainen tehtävä koostui SQL-kyselyn muodostamisesta ja opiskelijoiden tuli muodostaa kyselyt kolmella eri tavalla: puhtaasti SQL-kielellä, Keironin avulla ja Microsoftin Access -ympäristöllä.

Kaikki opiskelijat olivat saaneet opetusta näistä kolmesta järjestelmästä. Tutkimustulokset osoittavat, että Keiron pärjäsi hyvin yksinkertaisissa kyselyissä, joissa opiskelijan ei tarvinnut määritellä hakuehtoja. Se ei kuitenkaan noussut samalle tasolle muiden järjestelmien kanssa, kun kyselyt koskivat rajoittavia hakuja tai taulujen yhdistämistä. Aversanon tutkijaryhmä [2002] myöntää, että Keironiin kohdistuu teknisiä rajoitteita, jotka estävät sen täydellisen hyödyntämisen. Heidän mielestään ikoneihin perustuvalle järjestelmälle on kuitenkin oma paikkansa opetusjärjestelmien joukossa. [Aversano et al., 2002]

Myöskään GraphSQL Builder ei sisällä palautejärjestelmää tai muuta opiskelijaa tukevaa toiminnallisuutta. Siitä on kuitenkin tehty tutkimus, jolla pyrittiin selvittämään sitä, kuinka helposti ymmärrettäviä GraphSQL-kyselyt ovat. Testiin osallistui 10 testihenkilöä, jotka omasivat SQL:n perustiedot. Testi suoritettiin kahdessa osassa. Ensimmäisessä osassa testihenkilöille näytettiin neljä erilaista GraphSQL-kyselyä ja heitä pyydettiin muuntamaan kyselyt SQL-kyselyiksi. Näiden suoritusten suoritusaika mitattiin. Monimutkaisimman kyselyn tulokset vaihtelivat 16 sekunnista 68 sekuntiin. Testin tarkoituksena oli selvittää, pystytäänkö GraphSQL:ää käyttämään yhtä monipuolisesti kuin SQL-kyselyitä. Toisessa osassa testihenkilöt jaettiin kahteen ryhmään. Toiselle ryhmälle näytettiin GraphSQL-kyselyä ja toiselle ryhmälle vastaavaa SQL-kyselyä. Tämän jälkeen heitä pyydettiin vastaamaan muutamiin yksinkertaisiin kysymyksiin kyselyyn liittyen. Saatujen tulosten (kuva 12) perusteella GraphSQL tarjoaa nopeasti hahmotettavan ja selkeän esitystavan tietokantakyselyille. Testihenkilöiden määrä on kyseisessä testissä hyvin pieni, joten kovin merkittäviä johtopäätöksiä sen tuloksista ei voida tehdä. [Cerullo and Porta, 2010]



	Q1	Q2	Q3	Q4
SQL	80%	40%	80%	80%
GraphSQL	100%	80%	100%	80%

Kuva 12. GraphSQL:n hahmotustestin tulokset [Cerullo and Porta, 2010]

ADVICE:n palautekanavat eivät ole muihin järjestelmiin verrattuna kovin kattavia. Järjestelmä antaa opiskelijalle palautetta siitä, monellako rivillä esimerkkitulostaulu ja opiskelijan tekemän kyselyn tulostaulu erosivat toisistaan. Koska se on tarkoitettu käytettäväksi lähiopetuksessa, voi opettaja tarvittaessa opastaa opiskelijoita eteenpäin. Sen käytöstä ei ole vielä tehty käyttäjätestausta. Myöskään LEARN-SQL ei sisällä arkkitehtuurissaan palautekanavia. Koska

LEARN-SQL:ää ohjelmistoarkkitehtuurinaan käyttäviä järjestelmiä ei ole toteutettu vielä, ei siitä myöskään löydy käyttäjätestausmateriaalia.

3.4. Luokittelu

Arvioitujen ominaisuuksien perusteella tässä työssä käsitellyt järjestelmät on melko helppo luokitella kahteen eri luokkaan. Useimmat järjestelmistä ovat automatisoituja opetusjärjestelmiä, joiden käyttö onnistuu opiskelijalta myös omatoimisesti. SQLTutor, SQL-Tutor, Automated tutoring system ja ADVICE ovat tällaisia järjestelmiä. Niiden ominaisuuksiin kuuluvat mahdollisuus luoda opiskelijoille testejä, joista heidän on mahdollista saada palautetta automaattisesti. Opettajalle järjestelmät tarjoavat sähköisen kanavan tehtävien jakamiseen sekä mahdollisuuden seurata opiskelijoiden menestystä koko opetusjakson ajan. Käytän näistä järjestelmistä nimitystä *opetusjärjestelmä*.

Toisen ryhmän muodostavat Keiron ja GraphSQL. Ne ovat VQL-kieliin perustuvia työkaluja, joiden avulla käyttäjä pystyy luomaan SQL-kyselyitä visuaalisia elementtejä käyttäen. Tällaiset *visuaaliset työkalut* mahdollistavat omien VQL-kieliensä kääntämisen SQL-kielelle ja toisinpäin. Esitellyt järjestelmät eroavat toisistaan kuitenkin huomattavasti ja sopivat hyvin erilaiseen käyttöön. Aloittelevien käyttäjien kannattaa tutustua aluksi Keironiin, jonka syntaksi on GraphSQL:ää yksinkertaisempi. GraphSQL taas on edistyneiden käyttäjien työkalu, jonka avulla monimutkaisia kyselyitä on helpompi hahmottaa.

Viimeisen ryhmän muodostaa LEARN-SQL, joka ei itsessään ole valmis järjestelmä. Se on ohjelmistoarkkitehtuuri, jota noudattamalla voidaan rakentaa uusia opetusjärjestelmiä. Valmis LEARN-SQL:ään perustuva järjestelmä kuuluu *opetusjärjestelmien* kanssa samaan luokkaan.

4. Yhteenveto

SQL-kieltä ei enää mielletä tarpeeksi helppokäyttöiseksi, jotta se pystyisi täyttämään yhä kasvavan tietokantojen käyttötarpeen [Myers and Douglas, 2007]. Suurimmat ongelmat SQL-kielessä liittyvät sen syntaksiin [Brass and Goldberg, 2005], teorian soveltamiseen käytännössä [Hao et al., 2009a] ja tietokannanhallintajärjestelmien epätarkkaan virhemäärittelyyn [Brass and Goldberg, 2005]. Näiden ongelmien ratkaisuun on kehitetty useita erilaisia opetusjärjestelmiä. Tutkielmassani tarkastelin näitä järjestelmiä visuaalisuuden, automatisoinnin ja palautteenannon näkökulmista. Tarkastelun tuloksena järjestelmät voi luokitella kahteen eri luokkaan, *opetusjärjestelmiin* ja *visuaalisiin työkaluihin*.

Opetusjärjestelmillä pystytään antamaan täysvaltaista opetusta myös sellaisina aikoina, kun perinteinen luokkaopetus ei olisi mahdollista. Ne pystyvät opastamaan opiskelijoita oikeanlaisiin tehtäviin ja tarkastamaan näiden palauttamia tehtäviä automaattisesti. Vastauksiensa pohjalta opiskelijan on myös mahdollista saada automaattista palautetta. Opettajalle ne tarjoavat mahdolli-

suuden pysyä selvillä opiskelijoiden etenemisestä opetusjakson aikana. Tämän lisäksi ne tarjoavat normaalia SQL-ympäristöä muistuttavan ympäristön myös tenttitilaisuuksiin.

Visuaaliset työkalut toimivat opiskelijan tukena kyselyiden laatimisessa ja tulkitsemisessa visualisoimalla ne graafisella esitysmuodolla. Niiden avulla pystytään kääntämään visuaalisia kyselyitä SQL-kielelle tai tarvittaessa myös SQL-kyselyitä järjestelmän omaan esitysmuotoon. Tutkittavana olleet visuaaliset työkalut erosivat toisistaan merkittävästi ja ne osoittivat sen, että erilaiset visuaaliset työkalut voivat olla hyödyllisiä oppimisen eri vaiheissa.

Esitelty luokittelu ei ole täysin kattava vaan lähinnä luo pohjaa täydellisemmälle luokittelulle. Se ottaa kantaa tutkielmassa tarkasteltuihin ja arvioituihin ominaisuuksiin ja pyrkii tekemään suurpiirteisen jaon eri järjestelmien välillä. Luokittelua voidaan tarkentaa esimerkiksi luokittelemalla nyt esitellyt luokat aliluokkiin, jolloin päästään lähemmäksi kokonaisvaltaista luokittelua.

Viiteluettelo

[Abelló et al., 2008] Alberto Abelló, M. Elena Rodríguez, Toni Urpí, Xavier Burgués, M. José Casany, Carme Martín and Carme Quer, LEARN-SQL: Automatic assessment of SQL based on IMS QTI specification, In: *Proc. of Eighth IEEE International Conference on Advanced Learning Techonologies*, 2007, 592–593.

[Aversano et al., 2002] Lerina Aversano, Gerardo Canfora, Andrea De Lucia and Silvio Stefanucci, Understanding SQL through iconic interfaces, In: *Proc. of 26th Annual International Computer Software and Applications Conference*, 2002, 703–708.

[Brass and Goldberg, 2005] Stefan Brass and Christian Goldberg, Semantic errors in SQL queries: A quite compele list, *The Journal of Systems and Software* **79**, (2005), 630–644.

[Cerullo and Porta, 2007] Claudio Cerullo and Marco Porta, A system for database visual querying and query visualization: Complementing text and graphics to increase expressiveness, In: *Proc. of 18th International Workshop on Database and Expert Systems Applications*, 2007, 109–113.

[Cvetanovic et al., 2011] Milos Cvetanovic, Zaharije Radivojevic, Vladimir Blagojevic and Miroslac Bojovic, ADVICE – Educational system for teaching database courses, *IEEE Transactions on Education* **54**, 3 (2011), 398–409.

[Hao et al., 2009a] Yao-jun Hao, Jian-guo Wang and Qing-shan Zhao, The research on the technique of online experimental about SQL Tutor, In: *Proc. of Third International Symposium on Intelligent Information Technology Application*, 2009, 403–406.

[Hao et al. 2009b] Yao-jun Hao, Jian-guo Wang and Qing-shan Zhao, A dualistic and dynamic student model based on "Fuzzy Cluster", In: *Proc. of First*

International Workshop on Education Technology and Computer Science **1**, 2009, 693–696.

[Kenny and Pahl, 2005] Claire Kenny and Claus Pahl, Automated tutoring for a database skills training environment, In: Proc. of *SIGCSE '05*, 2005, 58–62.

[Mitrovic, 1998] Antonija Mitrovic, A knowledge-based teaching system for SQL, In: *Proc. of EDMEDIA'98 VA AACE* **98**, 1998, 1027–1032.

[Mitrovic and Martin, 2000] Antonija Mitrovic and Brent Martin, Evaluating the Effectiveness of Feedback in SQL-Tutor, In: *Proc. of International Workshop on Advanced Learning Technologies*, 2000, 143–144.

[Myers and Douglas, 2007] Colin Myers and Paul Douglas, The un-structured student, In: *Proc. of 24th British National Conference on Databases*, 2007, 3–9.

Assosiaatiosääntöjen klusterointi mielenkiintoisuuteen, etäisyyteen tai samankaltaisuuteen perustuen

Tuomas Neulaniemi

Tiivistelmä

Tässä tutkielmassa tarkastellaan ensiksi assosiaatiosääntöjä, niiden muodostamista sekä mittoja, joilla mitataan sääntöjen mielenkiintoisuutta. Assosiaatiosääntöjen määrää voidaan vähentää kiinnostavuusmittojen ja sääntöpeittojen avulla; tutkielmassa esitellään sääntöpeittoja tuottava algoritmi. Käsiteltävänä on myös klusterointi, jolla etsitään aineistossa olevia ryhmiä, joiden sisällä olevien tapausten välillä on mahdollisimman suuri samankaltaisuus. Lopuksi tarkastellaan menetelmiä, joilla lasketaan assosiaatiosääntöjen välinen etäisyys ja klusteroidaan assosiaatiosääntöjä.

Avainsanat ja -sanonnat: Assosiaatiosääntö, klusterointi, mielenkiintoisuus, samankaltaisuus, etäisyys, mitta, algoritmi.
CR-luokat: H.2.8, E.0

1. Johdanto

Assosiaatiosäännöt ovat ilmauksia, jotka koskevat aineistossa olevia säännönmukaisuuksia. Niitä käyttämällä voidaan löytää mielenkiintoisia yhteyksiä aineistossa olevien tietoalkioiden (items) (muuttuja-arvo -parien) välillä. Esimerkiksi assosiaatiosääntö {ostaa keskusyksikön = 1, ostaa näppäimistön = 1, ostaa näytön = 1} $\Rightarrow$ {ostaa tulostimen = 1} olisi sisällöllisesti seuraava: "Henkilöt, jotka ostavat keskusyksikön, näppäimistön ja näytön, ostavat yleensä myös tulostimen". Näitä neljää tuotetta ostetaan samanaikaisesti tietyssä osassa aineiston tapauksista. Sana yleensä on tietyn todennäköisyyden (suhteellisen osuuden) verrannollistuma. Säännössä vasemman puolen tietoalkiojoukko (itemset) koostuu yleensä yhdestä tai useammasta tietoalkiosta, kun taas oikealla puolella oleva tietoalkiojoukko koostuu yleensä ainoastaan yhdestä tietoalkiosta. Assosiaatiosäännön vasenta puolta kutsutaan säännön edeltäjäosaksi (antecedent) tai ehdoksi ja säännön oikeata puolta säännön seuraajaosaksi (consequent). Assosiaatiosääntö ei kuitenkaan ole käsitettävissä kausaliteetiksi, sillä säännön vasen puoli ei ole syy, eikä oikea puoli seuraus. Säännön luottamus (confidence) tarkoittaa, kuinka usein säännön oikea puoli esiintyy aineistossa säännön vasemman puolen esiintyessä. Tuki (support) ilmentää, miten usein

jokin tietty sääntö esiintyy havaitussa aineistossa. Assosiaatiosääntöjä käytetään aineistojen tutkimiseen ja luokitusten tekemiseen monilla sovellusalueilla, esimerkiksi bioinformatiikassa, tiedon louhinnassa Webissä ja ostoskorianalyysissa. [Han and Kamber, 2006].

Hanin ja Kamberin [2006] mukaan assosiaatiosääntöjen louhinnan tavoitteena on paikallistaa kaikki säännöt $X \Rightarrow Y$, joissa tuki ja luottamus ovat suurempia tai yhtä suuria kuin säännöille asetetut minimituki ja -luottamus. Assosiaatiosääntöjen louhinta on kaksivaiheinen prosessi, jonka ensimmäisessä vaiheessa on löydettävä kaikki kattavat (minimitukivaatimuksen täyttävät) joukot ja toisessa vaiheessa on muodostettava niistä assosiaatiosääntöjä. Ensin mainitun vaiheen laskenta on selvästi kalliimpaa kuin sääntöjen muodostaminen kyseisistä kattavista joukoista. Ensimmäiseen vaiheeseen on kehitetty useita algoritmeja, esimerkiksi Apriori-algoritmi, joiden avulla hakuavaruutta voidaan karsia tehokkaasti. Täten kattavat joukot voidaan löytää tehokkaasti. [Han and Kamber, 2006].

Assosiaatiosäännön mielenkiintoisuus tarkoittaa sitä, miten merkittävänä ja arvokkaana sääntöä voi pitää. Säännön mielenkiintoisuutta voidaan mitata kiinnostavuusmittareiden, esimerkiksi tuen ja luottamuksen, avulla. Generoitujen assosiaatiosääntöjen määrä on toisinaan liian suuri, vaikka erilaisia kiinnostavuusmittareita olisikin käytetty, ja siksi pyritään vähentämään sääntöjen määrää eri menetelmiä käyttäen. Sääntöjen määrän liiallisuus aiheuttaa resurssien ylikuormittumista. Kyseiseen ongelmaan on kehitetty ratkaisumalleja, joissa esimerkiksi karsitaan sääntöjä, jolloin poistetaan toistoa. Assosiaatiosääntöjen ryhmittelyä tarvitaan, sillä esimerkiksi sadan säännön manuaalinen tutkiminen vaatii runsaasti aikaa.

Klusterointi tarkoittaa menetelmää löytää aineistosta homogeenisia ryhmiä ja jakaa aineiston tapaukset kyseisiin ryhmiin, jolloin jokaisessa ryhmässä olevat tapaukset ovat keskenään mahdollisimman samankaltaisia. Tällöin ryhmät eli klusterit ovat keskenään erilaisia. Klusterointimenetelmien avulla voidaan ryhmitellä myös assosiaatiosääntöjä; tällöin ryhmiteltäviä tapauksia ovat assosiaatiosäännöt. On olemassa erityyppisiä klusterointimenetelmiä ja erilaisia metodeja määritellä tapausten välinen erilaisuus/etäisyys tai samankaltaisuus/läheisyys. Tässä tutkielmassa tarkastellaan assosiaatiosääntöjä, klusterointia sekä menetelmiä, joilla klusteroidaan assosiaatiosääntöjä. Lisäksi tarkastelun kohteena on assosiaatiosääntöjen ja assosiaatiosääntöklustereiden välisen etäisyyden/erilaisuuden mittaaminen. Luku 2 käsittelee assosiaatiosääntöjä. Kolmannessa luvussa esitellään klusterointia ja eräitä siinä käytettäviä menetelmiä. Luvussa 4 käsitellään assosiaatiosääntöjen klusterointia. Lopuksi esitetään yhteenveto käsitellyistä asioista.

2. Assosiaatiosääntö

2.1. Assosiaatiosäännöistä

Assosiaatiosääntö tarkoittaa datassa olevien määrättyjen tietoalkioiden samanaikaista esiintymistä koskevaa väitettä. Alkiojoukon kohdalla on olennaista, esiintyykö se kulloisessakin tilanteessa riittävän usein aineistossa. Tietoalkioiden määrää ei ole rajattu alkiojoukon sisällä. Assosiaatiosääntöön liittyy tuki, joka merkitään seuraavasti: T ($A \Rightarrow C$) = T($A \cup C$). Tuki ilmaisee, miten usein säännön vasen ja oikea puoli esiintyvät aineistossa yhdessä. Tuki voidaan ilmaista suhteellisena osuutena tai prosentuaalisesti. Luottamus määritellään seuraavasti: L ($A \Rightarrow C$) = P ($C \mid A$), jossa merkintä ($C \mid A$) tarkoittaa, että C esiintyy A:n ehdollistamana. Luottamus ilmaistaan prosentuaalisesti tai suhteellisena osuutena ja se kertoo, miten luotettava sääntö on. Mitä korkeampi prosenttimäärä on, niin sitä useammin oikea puoli esiintyy vasemman puolen esiintyessä ja sitä varmempi sääntö on. Merkittävä assosiaatiosääntöihin liittyvä käsite tuen ja luottamuksen lisäksi on noste, joka saadaan seuraavasti: luottamus ($A \Rightarrow C$) / tuki C. Noste kertoo, minkä suuruinen merkitys tapahtuman A esiintymisellä on tapahtuman C esiintymiseen. Luottamuksen ja säännön oikean puolen suhteellisen osuuden suhde muodostaa nosteen. Mikäli nosteen arvo on alle 1, niin säännön vasen ja oikea puoli korreloivat negatiivisesti. Mikäli arvo on 1, niin ne eivät korreloi keskenään. Mikäli nosteen arvo on yli 1, niin säännön vasen ja oikea puoli korreloivat positiivisesti. Noste voidaan laskea myös kaavalla

$$\text{noste}(A \Rightarrow C) = \frac{P(A \cup C)}{P(A)P(C)} . \qquad (2.1)$$

P($A \cup C$):lla tarkoitetaan A:n ja C:n havaittua yhdessä esiintymisen todennäköisyyttä ja nimittäjässä olevalla P(A)P(C):llä tarkoitetaan A:n ja C:n yhdessä esiintymisen todennäköisyyttä, jos A ja C ovat toisistaan riippumattomia. Mikäli nosteella on arvona 1, niin riippumattomuus pätee ja mikäli arvo on välillä 0-1, niin se ilmaisee suhdetta, joka ei ole niin voimakas kuin riippumattomuus. Arvon ollessa yli 1, riippuvuus on odotettua suurempaa. [Gray and Orlowska, 1998].

Seuraavaksi määritellään käsite ”kattava joukko”. Sääntöjen muodostamisessa tarvitaan joukkoja, jotka ovat kattavia (frequent itemsets). Joukko on kattava, mikäli säännölle asetettu minimituki täyttyy. Raja voi olla esimerkiksi 30%. Raja riippuu sovellusalueesta sekä subjektiivisesta päätöksestä. [Han and Kamber, 2006].

Assosiaatiosäännön vasemman puolen ehdot ovat käsitettävissä eräänlaisiksi eksistenssikvanttoreiksi, sillä jokaista ehtoa on olemassa jokin x eli olemassa oleva tapahtuma x. Universaalikvanttori olisi ajateltavissa seuraajaosaksi, koska jokainen edeltäjäosan ehto johtaa yhteen ja samaan seuraajaosaan, jolloin kyseinen seuraajaosa pätee kaikille juuri kyseiseen sääntöön kuuluville ehdoille x.

2.2 Esimerkkejä säännöistä

Kannan ja Bhaskaran [2009] esittelevät esimerkkejä säännöistä, joista mainittakoon

1. Temperature=cool & Humidity=normal 4 ⇒ Play=yes 3
tuki: 21%, luottamus: 75%.
2. Outlook=sunny & Humidity=high 3 ⇒ Play=no 3
tuki: 21%, luottamus: 100%.

Sääntö 1 tarkoittaa, että niissä tapauksissa, joissa lämpötila on viileä ja kosteus on normaali, niin vastaus pelaamiseen on kyllä. Sääntö 2 tarkoittaa, että mikäli ulkona on aurinkoista ja kosteus on korkea, niin vastaus pelaamiseen on ei. Heidän käyttämässään esimerkkiaineistossa on 14 tapausta. Säännössä 1 tuki 21 % tulee siten, että aineistossa olevia tapauksia on kaikkiaan 14 ja niistä kolme täsmää säännön vasempaan ja oikeaan puoleen, ja tällöin 3/14 * 100 ≈ 21 %. Luottamus 75 % tulee siten, että aineistossa on neljä tapausta, joissa säännön vasempaan puoleen kuuluvat Temperature on cool ja Humidity on normal. Näistä neljästä tapauksesta kolmessa säännön oikea puoli eli Play on yes, jolloin 3/4 * 100 = 75 %. Säännön 2 kohdalla tuki ja luottamus lasketaan vastaavalla tavalla. Noste säännölle 1 on 1.17, joka saadaan siten, että säännön luottamus jaetaan lasketulla säännön oikean puolen todennäköisyydellä (P(Play=yes)) ja täten lopulta lasketaan 75/64 ≈ 1.17. Säännölle 2 noste on 2.78, joka lasketaan vastaavalla tavalla. Esimerkiksi Temperature = ”cool” ja play = ”yes” ovat tietoalkioita. [Kannan and Bhaskaran, 2009].

2.3 Apriori-algoritmi

Pseudokoodina Apriori-algoritmi on esitetty algoritmissa 1.

C_k: candidate itemset of size k // C_k tarkoittaa kokoa k olevaa ehdokasjoukkoa
L_k : frequent itemset of size k // L_k tarkoittaa kokoa k olevaa kattavaa joukkoa

L_1 = {frequent items}; // tarkoittaa kokoa 1 olevia kattavia tietoalkiojoukkoja
for (k = 1; L_k != ∅; k++) do begin
C_{k+1} = candidates generated from L_k; // liitos (join) operaatio

```
    for each transaction t in database do
            increment the count of all candidates in Ck+1
        that are contained in t
    Lk+i  = candidates in Ck+1  with min_support
    end
return ∪kLk;  // palautetaan joukot, jotka ovat kokoa 1, 2, 3…
```

Algoritmi 1. Apriori

Apriori-algoritmilla voidaan louhia kattavia joukkoja tehokkaasti. Apriorin taustalla on periaate, jonka mukaan jokainen kattavan joukon osajoukko on myös kattava. Algoritmi etenee iteratiivisesti. Ensimmäiseksi algoritmi tutkii tietokannan etsimällä yhden kokoisia kattavia joukkoja. Seuraavaksi minimitukivaatimuksen täyttävistä tietoalkiojoukoista luodaan kokoa 2 olevat ehdokkaat, joille lasketaan tuki. Kullakin iteraatiokierroksella k algoritmi luo kokoa k+1 olevat ehdokkaat kokoa k olevista kattavista joukoista. Jotta ehdokasjoukko on potentiaalinen ehdokas, on sen jokaisen osajoukon oltava kattava. Tämän jälkeen algoritmi laskee kullekin kokoa k+1 olevalle ehdokasjoukolle sen tuen. Algoritmi päättyy, kun uusia kattavia joukkoja ei enää löydy tai kun uusien ehdokkaiden muodostaminen ei ole enää mahdollista. Apriori tekee täydellisen haun, jossa karsitaan hakuavaruutta tehokkaasti. [Han and Kamber, 2006].

2.4 ClusterApr-algoritmi

Kattavia joukkoja etsitään myös ClusterApr-algoritmilla, jonka pseudokoodi on esitetty Algoritmissa 2.

```
for all clusters M
    for all k > 2 and k ≤ |M|    // ehdokkaat, jotka ovat vähintään kokoa 2
        Insert each k-subset of M in Ck; // kukin k-alijoukko sijoitetaan hajautuspuuhun
for all transactions t ∈ D
        for all k > 2 and k ≤|t|
                for all k-subsets s of t  // kaikille ehdokkaiden k-alijoukoille
                    if (s ∈ Ck) s.count ++;      //  jos ehdokas ansaitsee paikan hajautus
                                                 //   puussa,  niin lisätään 1
        Lk = {c ∈ Ck|c.count ≥ minsup};   // minimituen täyttyessä ehdokasjoukko
                                          // lisätään kattavien joukkojen joukkoon
        Set of all frequent itemsets = ∪k Lk;   // yhdistetään kaikki löydetyt kattavat
                                                // joukot
```

Algoritmi 2. ClusterApr

M tarkoittaa klustereita, jotka sisältävät ehdokasalijoukkoja. Kaikki klusterit alijoukkoineen sijoitetaan hajautuspuihin, millä vältetään toisto eli kaksoiskappaleiden mukaanotto, ja varmistetaan ainutlaatuisuus. Algoritmi käy hajautuspuita läpi samalla karsimalla pois kaikki ne joukot, jotka eivät ole kattavia joukkoja. Käytettävissä on useita puita, aina yksi määrätyn mittaisille ehdokkaille. Ensin tuotetaan kaikki k:n mittaiset alijoukot. ClusterApr ylläpitää hajautuspuita laskeakseen tukia. Klustereita käsitellään yksi kerrallaan, alhaalta ylöspäin, kokoavalla menetelmällä. Algoritmin toiminta on siis kokoavaa ja jokainen klusteri käsitellään vuorollaan. ClusterAprin toiminta on hyvin samankaltaista kuin Apriorin kohdalla. Eroavuutena niiden välillä on se, että ClusterApr tutkii tietokannan ainoastaan kerran. [Zaki *et al*., 1997].

2.5 Joukkojen tyypeistä

Sääntöihin kytkeytyvät merkittävällä tavalla myös joukot, jotka voivat olla kattavien lisäksi suljettuja, suljettuja kattavia tai maksimaalisia kattavia joukkoja:

1. Suljetut joukot (closed itemsets). Joukko on suljettu, mikäli millään sen aidolla ylijoukolla ei ole samaa tukea kuin joukolla itsellään. Esimerkiksi joukon {b,c,d,e,f,g}, jolla on tuki 3, ylijoukko voisi olla {b,c,d,e,f,g,h}, jonka tuki olisi 2.
2. Suljetut kattavat joukot (closed frequent itemsets). Joukko on suljettu sekä kattava.
3. Maksimaaliset kattavat joukot (maximal frequent itemsets). Kattava joukko on maksimaalinen, mikäli yksikään sen aito ylijoukko ei ole kattava. [Han and Kamber, 2006].

Maksimaaliset kattavat joukot sisältävät suljetut kattavat joukot, jotka puolestaan sisältävät kattavat joukot. Suljetut kattavat joukot sekä maksimaaliset kattavat joukot ovat tavallaan subsumoitavissa kattaviin joukkoihin. Edellä kerrotut joukot ovat esitettävissä lattiisina, toisin sanoen hilaverkkona. Hilaverkossa on myös joukkoja, jotka eivät ole kattavia.

2.6 Säännön kompleksisuudesta

Säännön kompleksisuus on vähennettävissä säännön pituutta rajoittamalla ja assosiaatiosääntöjen määrän teoreettinen yläraja on rajattavissa. Huolimatta siitä, että sääntöjä saisikin rakennettua merkittävän selkeiksi, niin niiden määrä voi silti olla liian suuri. Käytettäessä korkeaa tukea mielenkiintoisuusmittana ongelmana on, että se todennäköisesti tuottaa lyhyempiä sääntöjä. Yhtenä ratkaisuna voisi olla vaihtelevan tuen käyttäminen: pidempiä sääntöjä voitaisiin löytää pienemmällä tuella. Mazlackin [2000] mukaan kannattaa kokeilla tapauskohtaisesti tukitasoa määrätylle datalle.

2.7 Sääntöjen peitto

Peitolla (cover) tarkoitetaan aineiston alkuperäisen sääntöjoukon kattavat tapaukset kattavaa sääntöjen alijoukkoa. Sääntöjen peitto on menetelmä löydettyjen sääntöjen karsimiseen. Löydetyissä sääntöjoukoissa on yleensä paljon toistoa, eli samoja aineiston rivejä luonnehditaan useilla säännöillä. Poistamalla toistoa saadaan sääntöjä vähennettyä. Ajateltaessa sääntökokoelmaa K, jonka kaikissa säännöissä on sama attribuuttijoukko Y seuraajana, kaikki säännöt kuvailevat attribuuttien Y:n läsnäoloa. Sääntöjoukon K alijoukko on sääntöpeitto, jos se kuvaa Y:tä aineiston kaikissa tapauksissa, joissa alkuperäinen sääntöjoukko kuvaa sen. Säännöiltä vaadittava korkea luottamus on sikäli kriittistä, että sen omaavat säännöt ovat mielenkiintoisia, mikä toimii seuraajalle Y oikeuttavana tekijänä. Tällöin sääntöjoukon ja peiton kokoa rajoitetaan. Jos halutaan kokonaistiedon säilyminen, on ajateltava, ettei ole pienemmän luottamuksen omaavaa erikoisemmaksi miellettävää sääntöä. Tällöin on tiedettävissä, ettei menetetä ristiriitaista informaatiota, vaikka pitäydytäänkin korkean luottamuksen omaavissa säännöissä. [Toivonen *et al.*, 1995].

2.8 Rakenteellinen sääntöpeitto

Toivonen ja kumppanit [1995] esittelevät myös rakenteellisen sääntöpeiton, joiden etsiminen toteuttaa karsimista nopeasti. Sääntöjoukko $\Delta \subseteq \Gamma$ on rakenteellinen sääntöpeitto Γ:lle, mikäli kaikille säännöille $(X \Rightarrow Y) \in \Delta$ ei ole olemassa sellaista sääntöä $(X' \Rightarrow Y) \in \Gamma$, että $X' \subset X$. Täten rakenteellinen sääntöpeitto sisältää alkuperäisen sääntöjoukon yleisimmät säännöt. Lähtökohtana on, että kaikille attribuuttijoukoille X,Y ja Z pätee, että $m(XYZ) \subseteq m(XZ)$, kun m(XYZ) ja m(XZ) tarkoittavat attribuuttijoukkoihin XYZ ja XZ täsmääviä rivijoukkoja. Eli säännön $X \Rightarrow Z$ täsmäämien rivien joukko on säännön $XY \Rightarrow Z$ täsmäämien rivien ylijoukko. Mikäli jälkimmäinen sääntö poistettaisiin sääntöjen peitosta, jäljelle jäävä sääntöjoukko olisi edelleen sääntöjen peitto. Jos säännöillä R_1: $FG \Rightarrow C$ ja R_2: $FGH \Rightarrow C$ on sama luottamus, niin jälkimmäinen sääntö on karsittavissa rakenteellisesta sääntöpeitosta, koska sillä ei ole enempää tietoa seuraajasta kuin edellisellä. [Toivonen *et al.*, 1995].

2.9 Rule cover -algoritmi

Rule cover -algoritmia käytetään parantamaan rakenteellisia peittoja, havaitsemaan lähellä optimaalisia olevia sääntöpeittoja sekä karsimaan sääntöjoukkoja. Kyseessä on heuristiikka, ahne menetelmä. Algoritmi varmistaa paikallisen optimaalisen sääntöpeiton. [Toivonen *et al.*, 1995.] Menetelmä esitellään algoritmissa 3.

Syöte: Sääntöjen joukko $\Gamma = \{ X_i \Rightarrow Y \mid i=1,\ldots,n\}$

Täsmäävien monikoiden/rivien joukko $m(X_iY)$ for all $i \in \{1,\ldots,n\}$

Tuloste: Sääntöpeitto Δ

Menetelmä:

$\Delta := \emptyset$;

$s' := \bigcup_{i=1}^{n} m(X_iY)$

for all $i \in \{1,\ldots,n\}$ do

    $s_i = m(X_iY)$;

end;

while $s' \neq \emptyset$ do

    valitse $i \in \{1,\ldots,n\}$ siten, että $(X_i \Rightarrow Y) \in \Gamma$ ja $|s_i|$ on suurin;

    $\Delta := \Delta \cup \{ X_i \Rightarrow Y \}$; // lisätään sääntö peittoon

    $\Gamma := \Gamma \setminus \{ X_i \Rightarrow Y \}$; // säännön poistaminen lähtöjoukosta

    for all $\{X_j \Rightarrow Y\} \in \Gamma$ do

        $s_j = s_j \setminus m(X_iY)$;

    end;

$s' := s' \setminus m(X_iY)$; // täsmäävät rivit poistetaan

end;

Algoritmi 3. Rule cover

Algoritmille annetaan alkuperäinen sääntöjoukko sekä sääntöjen täsmäävien rivien joukot. Peitto Δ alustetaan tyhjäksi joukoksi. Muuttujaan s' tallennetaan ne aineiston rivit, joihin mikään peiton säännöistä ei täsmää. Valitaan iteroiden sääntö $X_i \Rightarrow Y$. Tämän jälkeen kyseinen sääntö siirretään peittoon Δ. Jokaisella kierroksella valituksi tulee sääntö, joka peittää tähän mennessä peittämättömistä tapauksista suurimman määrän. Näitä vaiheita jatketaan, kunnes tapauksia ei enää ole jäljellä käsiteltäväksi. Algoritmin toiminta päättyy tilaan, jossa Δ sisältää alkuperäisen sääntöjoukon lähes minimaalisen sääntöpeiton. [Toivonen *et al.*, 1995]

2.10 Sääntöjen mielenkiintoisuus

Gray ja Orlowska [1998] toteavat, että tuki ja luottamus eivät ole riittäviä mielenkiintoisuusmittoja. Esimerkiksi sääntö $X \Rightarrow Y$, jossa tuki on 10 % ja luottamus 40 %, ei olisi riittävän mielenkiintoinen, mikäli seuraaja Y esiintyisi 40 %:ssa kaikista tapahtumista. He tuovat mielenkiintoisuuden käsitteeseen mukaan erottelun (discrimination), joka generoi viittauksen säännön edeltäjäosan ja seuraajaosan erillisyyksille. Generalisoidun assosiaatiosäännön $X_1 \Rightarrow Y_1$ erottelua merkitään kirjaimella d ja se määritellään kaavalla

$$d(X_1 \Rightarrow Y_1, r) = \frac{P(X1 \cap Y1)}{P(X1) * P(Y1)} \quad . \qquad (2.2)$$

Kaavan 2.2 avulla X_1:n ja Y_1:n riippumattomuus tulee esille. Kriteereinä säännön mielenkiintoisuudelle Gray ja Orlowska asettavat korkean tuen saamisen sekä erottelun, jolloin tuloksena säännön mielenkiintoisuuden määritykselle on kaava

$$i(X_1 \Rightarrow Y_1, r) = (d(X_1 \Rightarrow Y_1, r)^l - 1) * s(X_1 \Rightarrow Y_1)^m \quad . \qquad (2.3)$$

Kaavassa 2.3 olevia muuttujia l ja m käytetään parametreina punnitsemaan kahden mitan suhteellista merkittävyyttä. Kyseisten parametrien ollessa suurempia kuin 0 sekä termin $d(X_1 \Rightarrow Y_1, r)$ ollessa suurempi kuin 1 on seurauksena se, että $i(X_1 \Rightarrow Y_1, r)$ lisääntyy monotonisesti sekä $d(X_1 \Rightarrow Y_1, r)$:n että $s(X_1 \Rightarrow Y_1)$:n kanssa. Kaavassa r tarkoittaa monikkojoukkoa. Korkeamman mielenkiintoisuusarvon saavat säännöt ovat mielenkiintoisia. [Gray and Orlowska, 1998].

3. Klusterointi

3.1 Klusteroinnin luonteesta

Idea klusteroinnissa, jota kutsutaan myös klusterianalyysiksi, on löytää ryhmittelyyn perustuvilla menetelmillä samankaltaisia tapauksia aineistosta. Klusterointi on ohjaamatonta oppimista, koska luokkatietoa ei ole saatavilla, eli tiedolle, jota tulisi luokitella, ei ole tiedettävissä eksaktia luokkaa, vaan klusteroinnissa pyritään löytämään tapausten muodostamat luonnolliset luokat. Joskus on hankala määritellä klusteria, sillä esimerkiksi tapausten ja klustereiden eroavuudet eivät ole aina helposti todettavissa ja jokin koherentilta näyttävä klusteri saattaa olla tulkittavissa useammaksi kuin yhdeksi ryhmäksi. Mikäli tarkastelisimme klustereita kuvioina, niin klustereiden kesken vallitsevan eroavuuden voi luoda esimerkiksi pallon muoto, sakaroiden lukumäärä, jokin eroavuus yhdessä sakarassa tai etäisyyden vaihtelu. Klusterit voivat olla esimerkiksi toisistaan selvästi erottuvia tai keskipisteeseen perustuvia, jolloin tapaus sijoittuu siihen klusteriin, minkä keskipiste on sitä lähempänä. [Jain *et al.*, 1999]. Vierekkäisyyteen perustuva (contiguous) klusteri on sellainen pisteiden joukko, että klusterissa oleva piste on samankaltaisempi tai lähimpänä vähintään jotain toista samassa klusterissa olevaa pistettä siten, että kyseinen piste on selvästi lähempänä kuin muiden klusterien pisteet. [Tan *et al.*, 2006].

3.2 Samankaltaisuus ja erilaisuus

On tilanteita, jolloin ei ole yksiselitteistä määritellä samankaltaisuutta (similarity). Klustereista on tarkoitus muodostaa toisistaan selvästi erottuvia kokonaisuuksia. Tapauksiin kohdistuvaa samankaltaisuutta on määritelty monin tavoin. Samankaltaisiksi mielletyt korrelaatioita tuottamattomat tapaukset voivat merkitä, että klusteri on epätarkoituksenmukainen.

Erilaisuuteen (dissimilarity) tai etäisyyteen vaikuttaa tapauksiin kohdistuvan datan laatu, eli ovatko tapauksia kuvailevat muuttujat binäärisiä, nominaaleja, ordinaaleja, suhdeasteikollisia, välimatka-asteikollisia tai sekamuotoista dataa. Erilaisuus on binääristen muuttujien kohdalla mitattavissa siten, että muuttujilla on arvona 0 tai 1 ja näistä lasketut summat ilmentävät tapausten eroavuutta toisistaan. Välimatka-asteikollisten ja suhdeasteikollisten muuttujien kohdalla erilaisuus on mitattavissa esimerkiksi Manhattan-etäisyyttä tai euklidista etäisyyttä käyttämällä. Suhdeasteikollisten muuttujien kohdalla voi erilaisuus olla todettavissa myös siten, että niihin on sovellettavissa logaritminen muunnos ennen etäisyyksien laskemista. Sekamuotoisen datan kohdalla on käytettävä muuttujiin niiden mitta-asteikoiden mukaisia kaavoja. Vaikka muuttujia olisi siis montaa eri laatua, niin tapausten välinen erilaisuus on laskettavissa ja klusterit ovat siten muodostettavissa. [Han and Kamber, 2006].

3.3 Klusterointimenetelmiä

Klusterointialgoritmit voidaan jakaa hierarkkisiin, osittaviin, tiheysperustaisiin, taulukkoperustaisiin ja malliperustaisiin algoritmeihin. [Han and Kamber, 2006]. Tarkastellaan aluksi hierarkkisia klusterointimenetelmiä, jotka tuottavat klusterointihierarkian. Hierarkkinen klusterointi toimii kahdella tavalla, joista ensimmäisessä datapisteet yhdistetään klustereiksi ja toisessa klusterit jaetaan pienemmiksi klustereiksi.

Agglomeratiivinen eli kokoava klusterointi tarkoittaa alhaalta ylöspäin etenevää kokoamista. Aluksi kaikki tapaukset ovat omissa klustereissaan. Jokaisella tasolla tapahtuu sen klusteriparin yhdistäminen, jonka klustereiden välinen etäisyys on pienin. Lopuksi kaikki tapaukset kuuluvat samaan klusteriin.

Klustereiden välisen etäisyyden määrittäviä menetelmiä (link methods) on useita, esimerkiksi:

1. Yhden yhteyden menetelmä (single link), jossa kahden klusterin välinen etäisyys on niiden toisiaan lähimpänä olevien pisteiden välinen etäisyys (pienin etäisyys), joten päämääränä on etsiä lähin naapuri. Ensin todetaan pisteiden, joista toinen tulee klusterista x ja toinen klusterista y, välinen lyhin etäisyys. [Jain *et al.*, 1999].

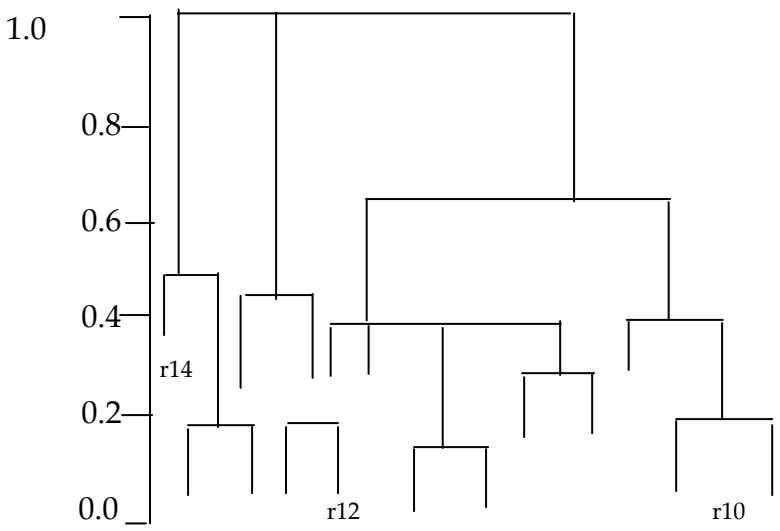
2. Keskimääräisen yhteyden menetelmä (average link), jossa kahden klusterin välinen etäisyys on niiden klustereiden pisteiden välinen keskimääräinen etäisyys. Tämä tarkoittaa naapurikeskiarvoa. [Han and Kamber, 2006].

3. Täydellisen yhteyden menetelmä (complete link), jossa kahden klusterin välinen etäisyys määritellään niiden toisistaan kauimpana olevien pisteiden välisenä etäisyytenä (suurin etäisyys), joten tavoiteltu naapuri on kauimpana oleva. [Jain *et al*., 1999].

Jakava klusterointi toimii päinvastaiseen suuntaan kuin kokoava. Kaikki tapaukset kuuluvat aluksi samaan klusteriin, minkä jälkeen yhdestä klusteriyhdisteestä puretaan jokaisella alaspäin etenevällä kierroksella klustereita pois, kunnes jokainen tapaus on oma erillinen klusterinsa. Kokoavaa klusterointia käytetään enemmän, koska jakava klusterointi on laskennallisesti raskasta. [Jain *et al*., 1999].

Klusteroinnin hierarkiaa kuvataan dendrogrammina, joka on eräänlainen rakenteellinen klusterointipuu. Puun lehdet ovat erillisiä tapauksia/pisteitä ja sisäsolmut edustavat klustereita. Dendrogrammista ilmenee klustereiden muodostumisjärjestys sekä se, mitkä klusterit ovat aliklustereita millekin klustereille. Dendrogrammi havainnollistaa aineiston struktuurin, yhdistämisvaiheessa vallinneen klusterien välisen etäisyyden sekä vaakatasolla olevat pisteet, jotka kertovat rakennetut klusterit. Havainnollistaminen voi helpottaa päättämään, mistä kohtaa klusterointipuu katkaistaan, toisin sanoen montako klusteria valittu klusterointi sisältää. Suuri muutos näissä yhdistämisetäisyyksissä voi olla mahdollinen katkaisukohta. [Jain *et al*., 1999]. Tapaukset olisivat merkittävissä numeraalisesti, mutta koska jo etäisyydet esitetään numeraalisesti, usein käytäntönä on merkitä tapaukset kirjaimilla.

Kuvassa 1 on esimerkki dendogrammista. Jokaisen alaspäin osoittavan viivan kohdalle kirjoitetaan kyseisen tapauksen (tässä säännön) numero, esimerkiksi r13, jossa r merkitsee sääntöä. Esimerkin vuoksi kuvassa on kolme säännön numeroa. Mitta-asteikko kuvaa klustereiden yhdistämisetäisyyttä.



Kuva 1. Esimerkki dendrogrammista

Osittavat algoritmit tuottavat klusterointihierarkian sijaan vain yhden osituksen eli klusteroinnin aineistosta. K:n keskiarvon algoritmi on osittava algoritmi, joka saa syötteenään halutun klustereiden lukumäärän k. K:n keskiarvon algoritmissa on tärkeänä elementtinä ryhmän keskipiste. Kukin havaintopiste sijoitetaan juuri siihen klusteriin, jonka keskipisteeseen havaintopisteestä on lyhin etäisyys. Algoritmi pysähtyy, kun klustereissa oleviin keskipisteisiin ei enää tule muutoksia. [Jain *et al*., 1999].

K:n keskiarvon algoritmi pyrkii minimoimaan neliövirhesummaa

$$E = \sum_{i=1}^{k} \sum_{x \in Ci} |x - mi|^2 . \qquad (3.1)$$

Kaavassa (3.1) x tarkoittaa vektoria, joka kuvaa datapistettä (tapausta) ja m_i tarkoittaa geometristä datapisteiden keskipistettä C_i:ssa. C_i tarkoittaa klusteria (tapausten osajoukkoa) ja k niiden määrää. Algoritmin aikakompleksisuus on O(t * k * n * p), kun t on tarvittavien iteraatioiden lukumäärä, k on klustereiden lukumäärä, n on attribuuttien lukumäärä ja p on pisteiden lukumäärä. K:n keskiarvon algoritmi on tehokas käsiteltäessä suuria datajoukkoja. [Han *et al*., 2001].

Tiheysperusteisessa klusteroinnissa (density-based) on tarkoituksena jakaa alkiot joukoiksi, joiden alkiot ovat mahdollisimman tiheässä. Klusterilla tarkoitetaan tahoa, jolla on suurempi pisteiden tiheys kuin sen ympärillä olevalla alueella. Klusteroituneiden pisteiden tiheys on ympäröivien pisteiden tiheyttä suurempi. Tällöin lähellä toisiaan olevat singulaariset pisteet luokitellaan samaan klusteriin. Klusterit ovat strukturoitavissa singulaaristen pisteiden attribuuttien perusteella. [Han *et al*., 2001]

Agrawal ja kumppanit [1998] esittelevät taulukkoperusteisessa (grid-based) klusteroinnissa käyttämänsä Clique-algoritmin, joka jakaa kutakin ulottuvuutta (muuttujaa) taulukoihin ja sen lisäksi laskee kaikissa ulottuvuuksissa olevat tiheät yksiköt. Seuraavaksi tiheät yksiköt yhdistetään tuottamaan tiheät yksiköt korkeammissa ulottuvuuksissa. Klusterit, jotka ovat hypersuorakulmioita, määritellään disjunktiivisilla normaalimuotoisilla (disjunctive normal form) lausekkeilla. [Agrawal, *et al.*, 1998].

Mallipohjainen klusterointi tarkoittaa menetelmiä, joissa yritetään optimoida sopivuutta annetun datan ja matemaattisen mallin välillä. Mainitut menetelmät perustuvat yleensä olettamukseen, että data on generoitu alla olevien todennäköisyysjakaumien sekoituksesta. Menetelmät noudattavat tilastollista lähestymistapaa tai neuroverkkoihin perustuvaa lähestymistapaa. [Han *et al.*, 2001].

Han ja kumppanit [2001] toteavat EM (expectation maximization) algoritmin esittävän jokaisen klusterin todennäköisyysjakaumaa käyttämällä. EM on iteratiivisesti toimiva metodi, jolla pyritään saavuttamaan samankaltaisuuden maksimi. Tarkoituksena on pyrkiä parantamaan klusterointilaatua alkuperäisestä ratkaisusta. Algoritmi laskee todennäköisyydet paikassa x sijaitsevalle tapaukselle kaavalla

$$P(x \mid i) = \frac{1}{\sqrt{(2\pi)^d |M_i|}} e^{\frac{1}{2}(x-\mu_i)^T (M_i)^{-1}(x-\mu_i)} \quad . \qquad (3.2)$$

Kaava (3.2) on multinormaalijakauman tiheysfunktio. M_i tarkoittaa kovarianssimatriisia ja merkintä $(M_i)^{-1}$ sen käänteismatriisia. $|M_i|$ on matriisi M_i:n determinantti. Matriisin transpoosi saadaan, kun matriisin rivit vaihdetaan sarakkeiksi ja sarakkeet riveiksi. Vektori μ merkitsee odotusarvoa, klusterin C_i keskiarvovektoria. [Han *et al.*, 2001].

4. Assosiaatiosääntöjen klusterointi

Assosiaatiosääntöjä voidaan karsia ja järjestää kiinnostavuusmittojen avulla. Luonnollisia mittoja mielenkiintoisuudelle ovat esimerkiksi tuki ja luottamus. Monimutkaisempia mittoja voidaan laskea tilastolliseen merkitsevyyteen perustuen. Kyseisten mittojen avulla mielenkiintoisimmat säännöt ovat löydettävissä, mutta mitat eivät auta suurten sääntökokoelmien esittämisessä. Sääntöjen määrää voidaan vähentää etsimällä sääntöjen peittoja, mutta peiton sisältämien sääntöjen määrä voi edelleen olla liian suuri. [Toivonen *et al.*, 1995]. Sääntöjoukon rakenteen muodostamisessa on hyödyllistä sääntöjen klusterointi. Kluste-

rointi auttaa ymmärtämään myös aineiston rakennetta, sillä se karakterisoi aineistoa. Tässä luvussa esitellään ensiksi mittoja, joilla voidaan mitata assosiaatiosääntöjen etäisyyttä sekä etäisyysmittoja hyödyntäviä mielenkiintoisuusmittoja. Lopuksi esitellään menetelmiä, joilla klusteroidaan assosiaatiosääntöjä.

4.1 Etäisyyksiä

Kahden säännön (r ja s) välillä voidaan käyttää kaavaa

$$\text{dist}(r,s) = 1 - \frac{\{\text{tietoalkiot r:ssä}\} \cap \{\text{tietoalkiot s:ssä}\}}{\{\text{tietoalkiot r:ssä}\} \cup \{\text{tietoalkiot s:ssä}\}} . \qquad (4.1)$$

Kaavaa käytetään binäärisen etäisyyden kohdalla. Kyseinen etäisyys sääntöjen välillä on verrattavissa niiden tietoalkioiden määrään, joiden esiintyminen on ainoastaan yhdessä säännössä tapahtuva. Suurta päällekkäisyyttä tietoalkioissaan omaavat säännöt on mahdollista asettaa samaan klusteriin. [Jorge, 2004].

Toivonen ja kumppanit määrittelevät sääntöjen $X \Rightarrow Z$ ja $Y \Rightarrow Z$ välisen etäisyyden rivien määräksi, jossa säännöt eivät ole ykseytyviä. He esittelevät kaavan

$$\begin{aligned} d(X \Rightarrow Z, Y \Rightarrow Z) &= |(m(X\,Z) \cup m(Y\,Z)) \setminus m(X\,Y\,Z)| \\ &= |m(X\,Z)| + |m(Y\,Z)| - 2|m(X\,Y\,Z)| \end{aligned} . \qquad (4.2)$$

Kaavassa $|\, m(x) \,|$ tarkoittaa täsmäävien rivien määrää, joka on etukäteen laskettu sääntöjen muodostamisprosessin aikana. [Toivonen *et al.*, 1995].

Gupta ja kumppanit [1999] esittelevät ehdolliseen todennäköisyyteen perustuvan etäisyysmitan kaavalla

$$\begin{aligned} d_{i,j} &= P\left(\overline{BS_i} \vee \overline{BS_j} \mid BS_i \vee BS_j\right) \\ &= 1 - \frac{\left|m\left(BS_i, BS_j\right)\right|}{\left|m(BS_i)\right| + \left|m(BS_j)\right| - \left|m(BS_i, BS_j)\right|}. \end{aligned} \qquad (4.3)$$

Kaavassa joukko BS_i on säännön i vasemman ja oikean puolen tietoalkioiden yhdiste. Kaavassa m tarkoittaa tietoalkioihin täsmääviä tapauksia. Kaavalla arvioidaan sääntöjen $R_i = B_i \Rightarrow S_i$ ja $R_j = B_j \Rightarrow S_j$ välistä etäisyyttä. Etäisyysmitalla on tarkoituksena ryhmitellä yhteen sääntöjä, joihin kohdistuu samanlainen tapauksien joukko.

Dong ja Li [1998] toteavat Toivosen ja kumppaneiden määrittelemässä semantiikkaperusteisessa etäisyydessä olevan vaarana naapuruston rakentumisen liian tiheäksi sekä vaikuttamisen naapurustoperusteiseen odottamattomuuteen. He määrittelevät syntaksiperusteisen etäisyyden, jossa on olennaista mitata sääntöjen välistä tietoalkioeroa, joka on jaettavissa kolmeen osaan. Osat ovat kaikkien tietoalkioiden symmetrinen ero kahdessa säännössä, vasempien puolien symmetrinen ero kahdessa säännössä sekä oikeiden puolten symmetrinen ero. Dongin ja Lin etäisyysfunktion tarkoitus on löytää odottamattomia sääntöjä erikseen määritellyllä naapurustolla. He määrittelevät yhden etäisyysmitoistaan kaavalla

$$Dist_{iset}(R_1, R_2) = \delta_1 * |(X_1 Y_1) \theta (X_2 Y_2)| + \delta_2 * |X_1 \theta X_2| + \delta_3 * |Y_1 \theta Y_2| . \quad (4.4)$$

Kaavalla (4.4) siis mitataan kahden säännön $R_1 = X_1 \Rightarrow Y_1$, $R_2 = X_2 \Rightarrow Y_2$ välistä etäisyyttä. δ_1, δ_2 ja δ_3 ovat termien suhteellista merkittävyyttä mittaavia parametreja ja iset tarkoittaa tietoalkiojoukkoa(item-set). Tarkemmin ottaen δ_1, δ_2 ja δ_3 ovat käyttäjän määritettävissä olevia painoja ja θ kuvaa symmetristä erilaisuutta X:n ja Y:n välillä. $|X|$ tarkoittaa X:n kardinaliteettia. [Dong and Li, 1998].

4.2 Mielenkiintoisuudesta

Mielenkiintoisuudella evaluoidaan assosiaatiosäännön tärkeyttä huomioiden säännön odottamattomuus (unexpectedness) [Dong and Li, 1998]. Säännön R_0 r-naapurustosta ($r > 0$) käytetään merkintää $N(R_0,r)$. Se on joukko $\{R: Dist_{iset}(R, R_0) \leq r$, R on potentiaalinen sääntö$\}$, jossa $r > 0$ on eräänlainen naapuruston säde. Säännön naapurusto kattaa määrätyllä etäisyydellä olevat säännöt. Sääntöjen välinen etäisyys lasketaan kaavalla 4.4. Naapurustoperusteisia mielenkiintoisuuksia ovat esimerkiksi odottamaton luottamus sekä odottamaton tiheys.

Odottamattomaan luottamukseen liittyy kaksi mittaa, jotka kuuluvat louhittujen sääntöjen luottamusten vaihteluun r-naapurustossa. Mitat ovat keskimääräinen luottamus ja luottamuksen keskihajonta. Voidaan ajatella, että M tarkoittaa louhittujen sääntöjen joukkoa, joka toteuttaa sille asetetun minimituen ja -luottamuksen sekä R_0 tarkoittaa louhittua sääntöä louhittujen sääntöjen joukossa M ja r on suurempi kuin 0. Tällöin mainitut mitat määritellään seuraavasti [Dong and Li, 1998]:

1. Sääntö R_0:n r-naapuruston keskimääräinen luottamus, joka määritellään sääntöjen luottamusten keskiarvoksi louhittujen sääntöjen joukossa $M \cap N(R_0,r) - \{R_0\}$; tämä arvo merkitään seuraavasti: keskimääräinen_luottamus(R_0,r).

2. Sääntö R_0:n r-naapuruston keskihajonta määritellään sääntöjen luottamusten keskihajonnaksi louhittujen sääntöjen joukossa $M \cap N(R_0,r) - \{R_0\}$; tämä arvo merkitään seuraavasti: keskihajonta_luottamus(R_0,r).

Kyseisen louhittujen sääntöjen joukon ollessa tyhjä nämä kaksi arvoa määritellään nollaksi, vaikka arvot ovat mahdollista määritellä muuksikin kuin nollaksi. Merkittävää on se, että sääntöjen luottamusten keskihajonnan arvo tuottaa luottamusten keskimääräisen vaihtelun säännön R_0:n r-naapurustossa.

Dong ja Li määrittelevät säännön R_0 r-naapurustossaan mielenkiintoiseksi (luottamukseltaan odottamattomaksi), mikäli seuraava toteutuu:
||luottamus(R_0)–keskimääräinenluottamus(R_0,r)–keskihajonta_luottamus(R_0,r)| on riittävän suuri, mikä on määritettävissä. Dong ja Li [1998] painottavat, että Luottamus(Ro):n erotuksen keskimääräisestä luottamus (Ro):stä on oltava merkittävästi suurempi kuin keskihajonta_luottamus(R_0,r).

Mielenkiintoisten sääntöjen saamiseksi on olemassa toinenkin tekniikka nimeltään harvat naapurustot. Siinä sääntöjen mielenkiintoisuutta tarkastellaan naapurustojen kohdalla, joissa on paljon potentiaalisia sääntöjä, mutta vain harvoja louhittuja sääntöjä. Harvin naapurusto on se, jolla on vähiten sääntöjä. Mikäli louhittujen sääntöjen määrä naapurustossa on selvästi pienempi kuin kaikkien naapurustossa olevien potentiaalisten sääntöjen määrä, niin tällöin sääntö on mielenkiintoinen. Mainittakoon, että Dong ja Li [1998] esittelevät kolmannenkin tekniikan, joka on sääntöjen kokoelmien mielenkiintoisuus.

Sääntöjen mielenkiintoisuutta (interestingness) voidaan mitata myös klusteroinnin avulla. Käytännön esimerkit ovat osoittaneet, että samankaltaiset tuotteet tai objektit eivät ole mielenkiintoisia, vaan niiden välillä vallitseva ei-samankaltaisuus toimii paremmin assosiaatiosääntöjen mielenkiintoisuuden mittarina. Ei-samankaltaisuuden laskemiselle on vaikeata antaa kaavaa, mutta klusterointi on siinä avuksi. Koska klusterointi pystyy erottamaan samankaltaiset objektit yhteen, niin sen avulla on tiedettävissä ei-samankaltaiset objektit. [Zhao, *et al.*, 2004].

Zhao ja muut [2004] esittävät osittain edelliseen perustuen määritelmiä mielenkiintoisuudelle, joista ensimmäisessä säännön $A \rightarrow B$ mielenkiintoisuus määritellään tietoalkion A klusterin ja tietoalkion B klusterin väliseksi etäisyydeksi. Kyseinen määritelmä sopii ainoastaan tapaukseen, jossa säännön vasemmalla puolella on ainoastaan yksi tietoalkio. Mutta koska tietoalkioita voi olla useampia, niin tarvitaan lisäksi määritelmä, jossa mielenkiintoisuus tulee määritellyksi klustereiden välisten etäisyyksien minimietäisyydeksi. [Zhao, *et al.*, 2004]. Jälkimmäinen kaava on kokonaisuudessaan seuraava:

$$\text{Interest}(A_1, A_2, \ldots, A_n \rightarrow B) = min_{i=1}^{n}\{\text{Dist}(C_{Ai}, C_B)\}. \qquad (4.5)$$

Käyttäjien vaatimukset ja määrätty skenaario ovat ratkaisevassa asemassa oikean klusterointialgoritmin valinnassa. Algoritmin tulisi kyetä käsittelemään sekamuotoista dataa, koska suurin osa assosiaatioanalyysin kohteena olevasta datasta on sekä numeerisia että kategorisia attribuutteja. Osittavat ja hierarkkiset algoritmit sopivat edellä käytettyyn lähestymistapaan. Tiheysperusteinen ja taulukkoperusteinen eivät auta klustereiden välisen ei-samankaltaisuuden toteamisessa, koska klustereita luodaan laajentamalla tiheästi asutettuja alueita. Sitä vastoin ei-samankaltaisuus on helposti laskettavissa osittavissa ja hierarkkisissa algoritmeissa. Zhao ja muut [2004] mainitsevat K:n keskiarvon algoritmin (K-means) yhtenä vaihtoehtona mitata sääntöjen mielenkiintoisuutta klusterointia käyttäen. He soveltavat algoritmia sekamuotoiseen dataan siten, että keskiarvoa käytetään numeerisiin attribuutteihin ja moodia kategorisiin attribuutteihin. Kaikkien kyseisten attribuuttien painotettu summa tulee määritellyksi kahden klusterin väliseksi etäisyydeksi. [Zhao, *et al.*, 2004].

4.3 Kaksiulotteisten assosiaatiosääntöjen klusterointi taulukkoperusteisella menetelmällä

Lent ja kumppanit [1997] esittelevät menetelmän, jossa klusteroidaan kaksiulotteisia assosiaatiosääntöjä. Säännöt noudattavat muotoa attribuutti1 & attribuutti2 $\Rightarrow$ C. Attribuutit 1 ja 2 ovat kvantitatiivisia ja C on kategorinen luokka-attribuutti. Heidän tekniikassaan on tärkeätä, miten yhdistetään kvantitatiivisten attribuuttien vierekkäiset intervallit (bin) tarkoituksen ollessa yleisempien sääntöjen ja entistä vähäisempien sääntöjen generointi. He esittelevät assosiaatiosääntöjen klusterointijärjestelmän (association rule clustering system), jossa on viisi komponenttia. Järjestelmässä on viisi merkittävää prosessia; intervallien muodostaminen (binner), assosiaatiosääntöjen etsiminen, sääntöjen klusterointi, sääntöklustereiden verifiointi ja lopuksi optimointi. Binner lukee aineiston tapauksia ja asettaa attribuuttien arvojen tilalle niitä vastaavien intervallien numerot. Seuraavaksi diskretoitu data siirtyy binneristä assosiaatiosääntökoneeseen, joka tuottaa assosiaatiosääntöjä, jotka siirtyvät klusterointikomponenttiin. Assosiaatiosääntö voisi olla esimerkiksi ikä(X,"30-31") & palkka(X,"100-200") $\Rightarrow$ ostaa (X,"Compaq tietokone"). Lent ja kumppanit kuvaavat tämän säännön kaksidimensionaalisessa taulukossa, jossa y-akselilla on palkka ja x-akselilla ikä. Akseleilla kuvataan siis säännön edeltäjäosassa olevia attribuutteja. Yhteen taulukkoon kootaan säännöt, joilla on samat attribuutit säännön vasemmalla ja oikealla puolella. Klusterointialgoritmi etsii klustereita näistä 2-ulotteisista taulukoista. [Lent *et al.*, 1997].

Klusteroitujen assosiaatiosääntöjen tarkkuutta testataan ja tarkkuus lähetetään heuristiselle optimoijalle, joka on viides komponentti. Optimoija säätää

minimitukikynnyksen ja/tai minimiluottamuskynnyksen, jonka jälkeen louhintaprosessi käynnistyy uudelleen assosiaatiosääntökoneesta. Säätöjä tehdään niin kauan kunnes sääntöjen tarkkuuden vahvistaja ei huomaa enää merkittävää parannusta tai asetettu aikaraja on saavutettu. [Lent *et al.*, 1997]

4.4 Assosiaatiosääntöjen tietoalkioperusteinen klusterointi

Jorge [2004] esittää tietoalkioperusteisen sääntöjen klusteroinnin, jossa on olennaista sääntöjen klusterointi niiden sisältämien tietoalkioiden perusteella. Tämän tuloksena tarjoutuu data-analyytikolle joukko sääntöjä, jotka vastaavat alueella olevia eri teemoja. Kyseinen temaattinen analyysi on mahdollista tehdä manuaalisesti edellyttäen, että sääntöjoukko on riittävän pieni sekä kukin sääntö vastaa erilaista teemaa. Esimerkkinä eri teemoista mainittakoon tilanne, jossa on neljä avainsanoja koskevaa sääntöä, jotka voivat käsitellä esimerkiksi vaalituloksia, kun taas toiset neljä sääntöä voivat käsitellä esimerkiksi erilaisia veroja. Vaikka säännöillä olisi yhteinen teema, niin säännöt voidaan ryhmitellä määrättyjen kriteerien mukaan. Sääntöjoukko on jaettavissa alijoukkoihin ja mikäli alijoukko on liian runsas, niin jakamista voidaan jatkaa kunnes kaikki alijoukot ovat halutun kokoisia. [Jorge, 2004]

Jorge [2004] määrittelee merkittävän temaattisen klusterin sisältävän sääntöjä, jotka jakavat toistensa kanssa tietoalkioita sillä erotuksella, että tietoalkioiden jakamista ei tapahdu muissa klustereissa olevien sääntöjen kanssa. Ensiksi on valittava sopiva etäisyysmetriikka sekä strategia klusterointia varten, sillä temaattista hyväksyttävyyttä ei muutoin saavutettaisi. Jorge käyttää tietoalkioiden päällekkäisyyteen perustuvaa etäisyysmittaa (kaava 4.1) ja hierarkkista kokoavaa klusterointia, jossa klustereiden väliset etäisyydet lasketaan keskimääräisen yhteyden menetelmällä.

Klustereiden määrän (klusterointipuun katkaisukohdan) valinta on merkittävässä asemassa, koska se vaikuttaa temaattisten alijoukkojen toisistaan erottuvuuteen, joka voi olla hyvinkin vaihtelevaa. Mikäli klustereiden määrä olisi satunnainen, niin ongelmalliseksi voi muodostua se, että johonkin klusteriin tulisi alijoukkoja, joilla ei olisi yhteisiä tapahtumia. [Jorge, 2004].

Jorge [2004] toteaa korkean temaattisen erillisyyden (distinctness) omaavien sääntöjen olevan erikoislaatuisia assosiaatioita omaavia sääntöjä. Erillisyysmitta on hyödyllinen tunnistamaan sääntöjä, jotka eivät ole muunnoksia toisista säännöistä. Säännön r erillisyys on 1, jos ja vain jos jokainen sääntö eroaa r:sta. Sääntöjoukossa R olevan säännön r erillisyyttä voidaan mitata kaavalla

$$\text{erillisyys}(r) = 1 - \frac{1}{\#R-1} \sum_{s \in R, s \neq r} (1 - \text{etäisyys}(r, s)) \; . \qquad (4.6)$$

Kaava (4.6) on laskettavissa etäisyysmatriisista. Mikäli kyseessä olisi suuria sääntöjoukkoja, niin saattaa olla kannattavaa huomioida säännön k lähimmät naapurit. [Jorge, 2004].

Sääntöjen klusterointi on parannettavissa iteratiivisesti sekä interaktiivisesti assosiaatiosääntöjen jälkikäsittely-ympäristössä PEAR:ssa (post processing environment of association rules). Ympäristö on web-pohjainen ja sen avulla on mahdollista tutkia sääntöjoukkoja sekä valita sääntöjä tai niiden synteesejä. Valittavissa on myös esimerkiksi operaattori, joka voi tuottaa säännön edeltäjäosan yleistyksen. Ohjelma generoi valintojen mukaiset säännöt, jotka ovat osin visualisoitavissa. [Jorge, 2004].

4.5 Assosiaatiosääntöjen klusterointi kokoavaa ketjuklusterointia käyttäen

Gupta ja kumppanit [1999] esittelevät algoritmin kokoavalle ketjuklusteroinnille. Algoritmi ei tarvitse koordinaattijärjestelmää (lähtökohtaisia datapisteitä), vaan kykenee löytämään klusterit käyttämällä ainoastaan tapausten välisiä etäisyysmittoja. Piste liitetään etäisyysmatriisista löytyvään lähimpään naapuriin ja tätä vaihetta toistetaan, kunnes kaikki pisteet on käyty läpi. Prosessi johtaa graafikokoelmaan. Kaikki samaksi graafiksi yhdistetyt pisteet saavat saman leiman. Algoritmi säätää klustereiden koot riippuen siitä, miten tiheässä pisteet ovat naapurustossa. Tiheä naapurusto johtaa kooltaan pieneen ja tiheään klusteriin. Tuloksena saadut klusterit ovat yksikäsitteisiä, eikä aloituspisteellä ole merkitystä. Algoritmin toiminta päätyy tilaan, jossa esitetään klusteroinnin eri tasoja kuvaava puurakenne. Algoritmi muistuttaa yhden yhteyden menetelmää, mutta erona on menetelmän tarkoitus tuottaa lyhyempiä klusteripuita ja kooltaan yhdenmukaisia klusteripuita. Toiminnaltaan algoritmi on seuraava: 1. Suorita ketjutus kaikille pisteille ja nouda kaikki klusterit. 2. Etsi jokaisen klusterin keskipiste. 3. Muodosta uusi avaruus, joka koostuu klustereiden keskipisteistä. 4. Mikäli keskipisteiden määrä on suurempi kuin 1, niin mene kohtaan 1. [Gupta et al., 1999].

Kokoava ketjuklusterointi toimii ulottuvuudettomasti: Ensimmäisessä vaiheessa pisteiden sijainti-informaatiota ei tarvita klusteroinnin toteutumisessa, vaan assosiaatiosääntöjen väliset etäisyydet lasketaan kaavalla 4.3. Toisessa vaiheessa riittää tieto klustereiden keskipisteiden välisestä etäisyydestä, joka on arvioitavissa useaa eri menetelmää käyttämällä ilman niiden koordinaatteja esimerkiksi kaavalla

$$d_{k(i,j)} = \alpha_i d_{ki} + \alpha_j d_{kj} + \beta d_{ij} + \gamma \left| d_{ki} - d_{kj} \right| . \qquad (4.7)$$

Kaavassa (4.7) oleva ryhmien etäisyyttä mittaava $d_{i,j}$ on eräänlainen funktio. Parametrilla β voidaan toteuttaa erilaisia klusterointiskeemoja.

5. Yhteenveto

Assosiaatiosääntöjen määrä on vähennettävissä kiinnostavuusmittojen, kuten tuen, luottamuksen ja nosteen avulla. Gray ja Orlowska sekä Dong ja Li ovat laajentaneet mittojen määrää generoidessaan omat mittansa, jotka perustuvat erotteluun ja sääntöjen odottamattomuuteen. Sääntöjen määrää voidaan vähentää myös etsimällä sääntöjen peittoja. Kiinnostavuusmittojen ja sääntöpeiton käytöstä huolimatta sääntöjen määrä voi olla edelleen liian suuri. Tällöin sääntöjoukon läpikäymistä ja ymmärtämistä voidaan helpottaa klusteroimalla sääntöjä. Tässä tutkielmassa on esitelty mittoja assosiaatiosääntöjen välisen etäisyyden laskemista varten ja menetelmiä, joilla klusteroidaan assosiaatiosääntöjä. Assosiaatiosääntöjen klusteroinnissa, kuten klusteroinnissa yleensäkin, sopivan etäisyysmitan ja klusterointimenetelmän löytämisen merkitys on vaikeasti yliarvioitavissa.

Viiteluettelo

[Agrawal, *et al.*, 1998] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos and Prabhakar Raghavan, Automatic subspace clustering of high dimensional data for data mining applications. In: *Proc. of 1998 ACM SIGMOD Int. Conf. on Management of Data*, 94-105.

[Dong and Li, 1998] Guozhu Dong and Jinyan Li, Interestingness of discovered association rules in terms of neighborhood-based unexpectedness. In: *Proc. of 2nd PAKDD*, (1998), Springer-Verlag, 72-86.

[Du *et al.*,1999] Xiaoyong Du, Sachiko Suzuki and Naohiro Ishi, A distance-based clustering and selection of association rules on numeric attributes. In: *Proc. of New Directions in Rough Sets, Data Mining, and Granular-Soft Computing, Lecture Notes in Computer Science* **1711** (1999), 423-432.

[Gray and Orlowska, 1998] Brett Gray and Maria E. Orlowska, CCAIIA: Clustering categorical attributes into interesting association rules. In: *Proc. of Research and Development in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science* **1394**/(1998), Springer-Verlag, 132-143.

[Gupta *et al.*, 1999] Gunjan. K. Gupta, Alexander Strehl and Joydeep Ghosh, Distance-based clustering of association rules. In: *Proc. of Intelligent Engineering Systems through Artificial Neural Networks*, (ANNIE 1999), 759-764.

[Han *et al.*, 1997] Eui-Hong Han, George Karypis, Vipin Kumar and Bamshad Mobasher. Clustering based on association rule hypergraphs. In: *Proc. of*

SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, (1997), 1-22.

[Han *et al*., 2001] Jiawei Han, Micheline Kamber and Anthony K. H. Tung, Spatial clustering methods in datamining. In: *Geographic Data Mining and Knowledge Discovery, Taylor and Francis,* (2001), 1–29.

[Han and Kamber, 2006] Jiawei Han and Micheline Kamber, *Data Mining: Concepts and Techniques*, 2nd ed. Morgan Kaufmann Publishers, (2006).

[Jain *et al*., 1999] Anil Kumar Jain, M Narasimha Murty, Patrick Joseph Flynn. Data Clustering: A Review. *ACM Computing Surveys* **31**(3), (1999), 264-323.

[Jorge, 2004] Alipio Jorge, Hierarchical clustering for thematic browsing and summarization of large sets of association rules. In: *Proc. of the Fourth SIAM International Conference on Data Mining*, (2004), 178-187.

[Kannan and Bhaskaran, 2009] S. Kannan and R. Bhaskaran. Association rule pruning based on interestingness measures with clustering. *Journal of Computer Science* **6**, (2009), 35-43.

[Lent *et al*., 1996] Brian Lent, Arun Swami and Jennifer Widom, Clustering association rules. In: *Proc. of the Thirteenth International Conference on Data Engineering IEEE Computer Society*, (1997), 220-231.

[Mazlack, 2000] Lawrence J. Mazlack. Approximate clustering in association rules. In: *Proc. of the 19th International Conference of the North American Fuzzy Information Processing Society, 2000,* 256-260.

[Tan *et al*., 2006] Pang-Ning Tan, Michael Steinbach and Vipin Kumar, *Introduction to Data Mining*, Addison-Wesley, (2006).

[Toivonen *et al*., 1995] Hannu Toivonen, Mika Klemettinen, Pirjo Ronkainen, Kimmo Hätönen and Heikki Mannila, Pruning and grouping discovered association rules. In: *Proc. of the MLnet Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases,* (1995), 47-52.

[Zaki *et al*., 1997] Mohammed J Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, New algorithms for fast discovery of association rules. In: *Proc. of the 3rd Intl Conf on Knowledge Discovery and Data Mining* **20** (651), AAAI Press, (1997), 283-286.

[Zhao *et al*., 2004] Yanchang Zhao, Chengqi Zhang and Shichao Zhang, Discovering interesting association rules by clustering. In: *Proc. of the 17th Australian Joint Conference on Artificial Intelligence*, (2004), 1055-1061.

Uutisten personointi

Tuuli Paananen

Tiivistelmä

Uutisten personoinnilla tarkoitetaan tässä tutkielmassa sitä, että uutispalvelu tarjoaa lukijalle ensisijaisesti sellaisia uutisia, jotka häntä kiinnostavat. Tämä voidaan toteuttaa erilaisia ohjelmistoja ja algoritmeja käyttäen. Tutkielmassa luodaan aluksi katsaus uutisten personoinnin tekniikkaan, minkä jälkeen pohditaan arvonäkökulmia journalismin kannalta.

Avainsanat ja -sanonnat: Uutisten personointi, sanomalehdet, verkkolehdet, journalismi, käyttäjäkokemus

CR-luokat: H.5

1. Johdanto

Uutisten lukeminen ja vastaanottaminen ovat muuttuneet aivan ensimmäisistä lähettien mukana kuljetetuista tiedonannoista, sanomalehden, radio- ja televisiolähetysten kautta verkkouutisointiin. Internetissä sisällön ja palveluntarjoajien lukumäärä on todella suuri, jolloin on tarpeellista auttaa lukijaa löytämään kiinnostavimmat artikkelit. Informaation suodatus ja sisällön personointi käyttäjän mielenkiinnon kohteiden mukaisiksi ovat eräs vastaus tähän informaatioähkyyn [Liu et al., 2010]. Personointi on mahdollista toteuttaa joko eksplisiittisesti käyttäjän aktiivisia valintoja vaativalla tavalla tai implisiittisesti käyttäjän valintoja seuraamalla [Thurman, 2011]. Uutisten personoinnin voidaan nähdä siirtävän yhteiskunnan portinvartijan (gate-keeping) vastuun journalisteilta tietokoneohjelmien kautta uutisten lukijoille.

Tilastokeskuksen [SVT 2009, 2010] julkistamien tietojen perusteella verkkomedia vie tilaa paperiselta sanomalehdeltä. Vuosina 2008–2009 sanomalehtien levikki pieneni enemmän kuin koskaan sotien jälkeen: pudotus oli 10 prosenttia. Samaan aikaan verkkomedia Internetissä kasvoi 5 prosenttia. Vuosina 2009–2010 sanomalehtien tilanne näytti hieman valoisammalta, kun nousua edellisvuoteen oli 2 prosenttia. Verkkomedia jatkoi kuitenkin kasvuaan merkittävällä 15 prosentilla. Keskittyminen uutistoimistojen verkkopalveluiden kehittämiseen personoinnin avulla tuntuu tässä valossa luontevalta journalismibisneksen pelastusyritykseltä. Uutisten personointiin liittyy kuitenkin useita arvokysy-

myksiä tiedon rajautumisesta paperisen sanomalehden tulevaisuuteen. Näihin kysymyksiin pyrin tässä tutkielmassa vastaamaan.

Tutkielman rakenne on seuraavanlainen. Ensin esittelen erilaisia uutisten personoinnin tapoja jakaen ne eksplisiittiseen ja implisiittiseen personointiin. Rajaan kuitenkin personoinnin matemaattisen esityksen tutkielmani ulkopuolelle. Kolmannessa luvussa esittelen tarkemmin kaksi uutisten personointia hyödyntävää verkkosovellusta, Krakatoa Chroniclen ja Google Newsin. Tämän jälkeen pohdin uutisten personointia käyttäjäkokemuksen näkökulmasta. Viidennessä luvussa paneudun uutisten personoinnin arvonäkökulmaan pohtien ensimmäiseksi portinvartijateoriaa. Koetan myös selvittää, mitä vaikutusta uutisten personoinnilla on journalistin työn kannalta. Viimeisessä luvussa teen yhteenvedon tutkielmani tuloksista.

2. Uutisten personoinnin toteutus

Tässä tutkielmassa jaan uutisten personoinnin toteutuksen kahteen luokkaan sen mukaan, miten käyttäjän mielenkiinnon kohteet on huomioitu. Ensimmäinen on eksplisiittinen toteutus, joka hyödyntää suoraan käyttäjältä saatuja syötteitä. Implisiittinen toteutus sen sijaan perustuu erilaisiin algoritmeihin. Esittelen myös PNS-mallin, jonka avulla on mahdollista kuvata uutisten personoinnin teknologista toimintaperiaatetta.

2.1. Uutisten personoinnin eksplisiittinen toteutus

Eksplisiittinen personointi toimii käyttäjän aktiivisten valintojen perusteella, jolloin käyttäjä pystyy näkemään tekemänsä valinnat ja halutessaan myös muuttamaan niitä myöhemmin. Taulukossa 1 olen kuvannut Thurmanin [2011] esittelemän jaottelun eksplisiittisen personoinnin kategorioista.

Kategoria	**Määritelmä**
Sähköinen uutiskirje (Email Newsletter)	Rekisteröityessään järjestelmään käyttäjä valitsee haluamansa asetukset, joihin kuuluvat *formaatti* (HTML tai tavallinen teksti), *toimitusaikataulu* (esim. päivittäin tai viikoittain), rekisteröinnin yhteydessä asetetut *uutiskategoriat* ja *avainsanat*.
Tekstiviestit (SMS alerts)	Käyttäjä rekisteröityy järjestelmään ja asettaa haluamansa avainsanat tai -kategoriat, joiden perusteella hänelle lähetään tekstiviestiuutisia kiinnostavista aiheista.
Etusivun personointi (Homepage Customization)	Käyttäjä voi tehdä omat sisältö- ja ulkoasuasetukset uutispalvelun etusivulle.

Vaihtoehtoiset etusivut (Homepage Editions)	Käyttäjä voi valita vaihtoehtoisen etusivun. Valinta on yleensä binaarinen: käyttäjä voi valita esimerkiksi kansallisen tai alueellisen etusivun. Etusivun personoinnista tämä eroaa valintojen moninaisuudessa. Tässä käyttäjällä on vain hyvin rajallinen määrä vaihtoehtoja.
Oma sivu (My Page)	Käyttäjän oma henkilökohtainen sivu uutispalvelun verkkosivustolla (muu kuin etusivu). Omalle sivulle voi sisällyttää erilaisia sivuston sisäisiä moduuleja sekä ulkoista sisältöä RSS-syötteiden avulla. Lisäksi ulkoasua pystyy muokkaamaan.
Tarinani (My Stories)	Käyttäjä voi tallentaa suosikkiartikkelejaan omalle sivulleen lehtileikkeiden tapaan.
Mobiiliuutiset (Mobile Editions)	Henkilökohtaiset asetukset uutissivustojen mobiiliversioihin. Mahdollisia muuttujia ovat muun muassa ulkoiset RSS-syötteet, käyttäjän määrittämä sijainti ja avainsanat. Lisäksi käyttäjällä on mahdollisuus tallentaa artikkeleja myöhempää käyttöä varten.
Epälineaariset vuorovaikutustavat (Non-linear Interactives)	Sisäänrakennetut sovellukset, jotka välittömästi mukauttavat sisällön ja esitystavan käyttäjän valintojen mukaiseksi. Liittyy yleensä johonkin tiettyyn uutistapahtumaan.
Kahdenkeskinen yhteistoiminnallinen suodatus (One-to-one Collaborative Filtering)	Käyttäjä vastaanottaa sisällöllisiä suosituksia yksittäisiltä toimittajilta tai muilta käyttäjiltä.
RSS-syötteet (RSS-feeds)	Mahdollisuus tilata RSS-syöte. Muuttujia ovat formaatti, pituusrajoitukset sekä personointi avainsanojen ja kategorioiden avulla.
Pienoisohjelmat (Widgets)	Käyttäjä voi ladata esimerkiksi tietokoneensa työpöydälle pienoisohjelmia, joiden kautta hän pystyy lukemaan kiinnostavat uutiset.
Muut eksplisiittiset (Other Explicit)	Kaikki se uutissivuston mukautuminen sisällön, jakelutavan tai uutisten järjestyksen osalta käyttäjän mieltymysten mukaiseksi, jota ei ole mainittu muualla tässä taulukossa.

Taulukko 1: Eksplisiittisen personoinnin toiminnallisuudet uutissivustoilla [Thurman, 2011]

Taulukon 1 kategorioita voi toiminnallisuuden perusteella niputtaa yhteen. Sähköinen uutiskirje ja tekstiviestit ovat kumpikin pääasiassa sivuston ulko-

puolista toimintaa. Käyttäjä on ensin uutissivustolla valinnut haluavansa tilata sähköisen uutiskirjeen tai tekstiviestiuutiset, minkä jälkeen hän asettaa tietyt ehdot niiden lähettämiselle. Näissä kummassakin ideana on kuitenkin se, että uutinen tulee joko sähköpostiin tai matkapuhelimeen, josta se on luettavissa kokonaisuudessaan.

Varsinaisen verkkosivun personointiin kuuluvat kategoriat *etusivun personointi*, *vaihtoehtoiset etusivut*, *oma sivu* ja *tarinani*. Näihin kaikkiin neljään kategoriaan kuuluu jonkinlainen mahdollisuus mukauttaa uutissivustoa käyttäjän mielihalujen mukaiseksi. Näihin kuuluu niin ulkoasun kuin sisällönkin mukauttaminen. Myös mobiiliuutiset vaikuttavat varsinaiseen uutissivustoon, joskin vain sen mobiiliversioon.

Thurmanin [2011] mukaan uutisten personoinnin eksplisiittinen toteutus vaatii käyttäjältä paljon aikaa ja vaivaa. Niillä uutissivustoilla, joilla eksplisiittinen personointi on mahdollista, on sen käyttö ollut hyvin vähäistä. Useat tutkijat ovatkin sitä mieltä, että käyttäjät ovat melko passiivisia uutisten lukemistavoissaan, jolloin implisiittinen uutisten personointi on useimmille mieluisampi vaihtoehto.

2.2. Uutisten personoinnin implisiittinen toteutus

Implisiittinen toteutus perustuu erilaisiin algoritmeihin. Implisiittistä personointia käyttävä järjestelmä seuraa käyttäjän lukemia uutisia ja niiden aihealueita, eikä tämä vaadi käyttäjältä muita toimenpiteitä palveluun rekisteröitymistä lukuun ottamatta. Tämä tarkoittaa sitä, että uutistarjonnan mukautuminen käyttäjän mielenkiintojen mukaiseksi ei vaadi tältä minkäänlaisia aktiivisia valintoja. Taulukossa 2 olen kuvannut Thurmanin [2011] esittelemän jaottelun implisiittisen personoinnin kategorioista.

Kategoria	**Määritelmä**
Yhteistoiminnallinen suodatus (Aggregated Collaborative Filtering)	Uutissisällöt määräytyvät automaattisesti niiden suosion mukaan. Muuttujiin kuuluvat luetuin, katsotuin, etsityin ja jaetuin uutinen. Valintoja voidaan vielä tarkentaa uutisaiheiden *kategorian* (esim. urheilu), *julkaisuajan* (esim. viimeisten 24 tunnin sisällä) ja käyttäjän maantieteellisen *sijainnin* (esim. Suomi tai Eurooppa) mukaan.
Kontekstuaaliset suositukset (Contextual Recommendations)	Järjestelmä suosittelee asiayhteyteen liittyvää sisältöä, joka voi olla artikkeli, blogi, video tai valokuva. Suositeltu sisältö ei välttämättä ole samalla sivustolla vaan se voi myös sijaita jossakin muussa osoitteessa.
Maantieteellinen kohdentaminen	Sisältö mukautuu käyttäjän maantieteellisen sijainnin mukaan, joka voidaan selvittää IP-osoitteen tai jonkin

(Geo-targeted Editions)	muun menetelmän avulla.
Käyttäjäprofiiliin perustuvat suositukset (Profile-based Recommendations)	Sisältösuositukset perustuvat käyttäjäprofiiliin. Profiili voidaan tehdä joko järjestelmän sisäisesti käyttäjärekisteriin syötettyjen tietojen tai tallentamalla käyttäjän käyttäytymistä sivustolla (esim. klikkauskäytännön seuraaminen) tai tuomalla informaatio muilta organisaatioilta kuten sosiaalisen median sivustoilta.

Taulukko 2: Implisiittisen personoinnin toiminnallisuudet uutissivustoilla [Thurman, 2011]

Kuten taulukosta 2 nähdään, implisiittinen personointi voidaan toteuttaa erilaisilla tavoilla. Järjestelmän kannalta helpoin on yhteistoiminnallinen suodatus, jossa uutisotsikoiden järjestys määräytyy niiden suosion mukaan. Lisäksi käyttäjä voi tehdä tarkentavia rajauksia julkaisuajankohdan, kategorian tai maantieteellisen sijainnin mukaan. Tällä tavalla ei kuitenkaan saada aikaiseksi täysin persoonallista, eli jokaiselle käyttäjälle henkilökohtaista, sivustoa.

Kontekstuaalisilla suosituksilla päästään jo lähemmäksi aidosti henkilökohtaista uutissivustoa. Kun käyttäjä on valinnut jonkin artikkelin luettavakseen tarkemmin, järjestelmä suosittelee samaan aihepiiriin liittyviä sisältöjä. Suositeltu sisältö ei välttämättä sijaitse kyseessä olevalla uutissivustolla, vaan se voi olla myös jossakin muussa osoitteessa.

Eksaktein ja näistä neljästä mielenkiintoisin on käyttäjäprofiiliin perustuvat suositukset. Käyttäjän tulee rekisteröityä sivustolle, minkä yhteydessä hän voi tehdä eksplisiittisesti joitakin sisältöön liittyviä valintoja esimerkiksi uutiskategorian mukaan. Lisäksi järjestelmä seuraa automaattisesti käyttäjän klikkauskäytäntöä, eli minkä tyylisiä uutisia käyttäjä lukee.

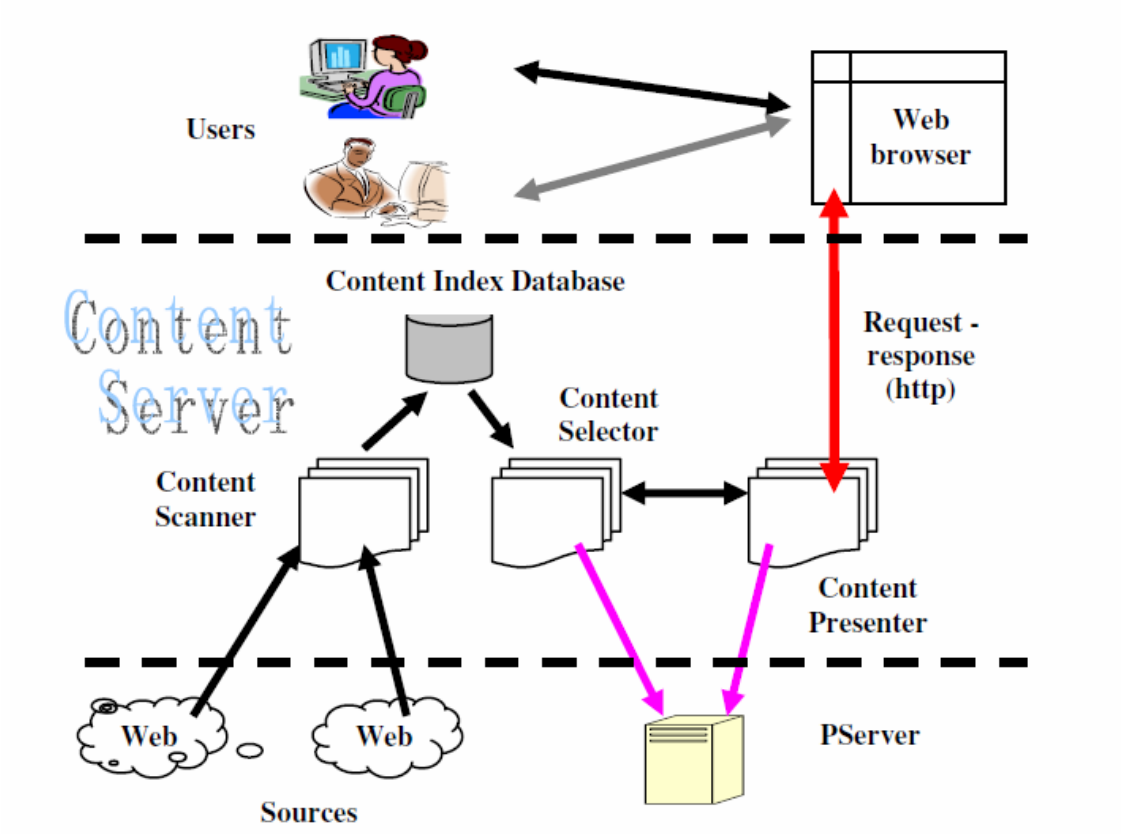
Implisiittinen personointi on osoittautunut taloudellisesti kannattavaksi: ensinnäkin koska algoritmi on halvempi kuin toimittaja, ja toiseksi personoinnin avulla on mahdollista kohdentaa mainonta entistä paremmin käyttäjästä kerätyn tiedon avulla. Implisiittiseen personointiin liittyy kuitenkin useita arvokysymyksiä. Implisiittisen personoinnin mekanismit ovat vaikeasti havainnoitavissa, koska ne eivät vaadi käyttäjältä aktiivista osallistumista, ja koska ne perustuvat monimutkaisiin algoritmeihin. [Thurman, 2011.]

2.3. Uutisten personoinnin PNS-malli

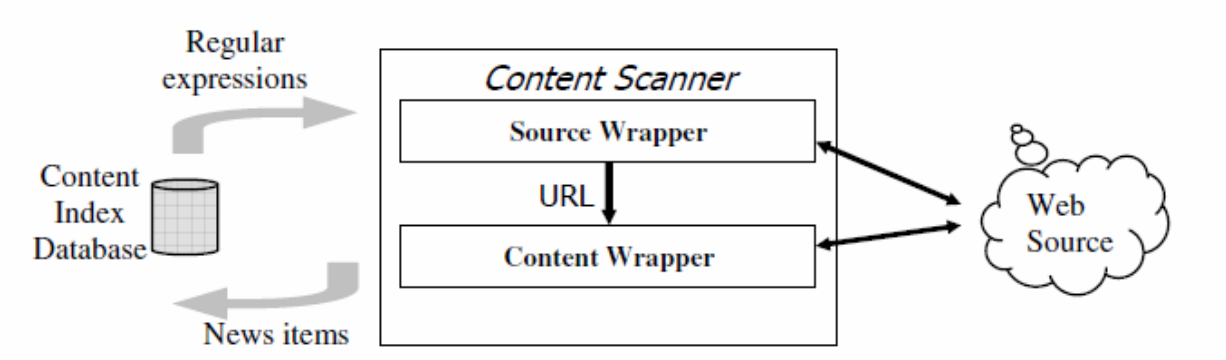
Paliouras ja muut [2006] esittelevät uutisten personoinnin PNS-mallin. Se tarjoaa käyttäjälle mahdollisuuden lukea personoituja uutisaiheita, jotka on kerätty useista eri lähteistä. PNS-sisältöpalvelimen avulla on mahdollista tuottaa henkilökohtainen sähköinen sanomalehti, joka vastaa käyttäjän kiinnostuksen kohtei-

ta. Sisältö muodostuu implisiittisesti käyttäjän lukuhistorian ja muiden käyttäjäprofiileiltaan lähellä olevien käyttäjien suositusten perusteella.

Kuvassa 1 esitellään järjestelmän toimintaperiaate. Se koostuu kolmesta pääkomponentista: *sisällön suodattajasta* (content scanner), *sisällön valitsijasta* (content selector) ja *sisällön esittäjästä* (content presenter). Sisällön suodattaja ja sisällön valitsija kommunikoivat hakemistotietokannan (content index database) kanssa, jonne tallennetaan kaikki relevanteimmat uutisaiheet. Sisällön suodattaja on itsenäinen moduuli, joka hakee uutisaiheita eri sisältölähteistä tietyin aikavälein käyttäen tiedonlouhintatekniikkkaa. Tätä toimintaa on kuvattu tarkemmin kuvassa 2. Sisällön suodatus tapahtuu kahdella tasolla, jolloin tarvitaan myös kaksi käärettä (wrapper). Ylemmän tason kääre (source wrapper) tuo luetuimpien artikkeleiden URL-osoitteet uutislähteen etusivulta. URL lähetään alemman taso kääreelle (content wrapper), joka tallentaa ne tietokantaan.



Kuva 1. PNS-malli [Paliouras et al., 2006]



Kuva 2. Uutissisällön suodattajan rakenne [Paliouras et al., 2006]

Järjestelmä kerää tietoa käyttäjistä rekisteröitymisen avulla. Sitä varten tarvitaan käyttäjänimi ja salasana. Tämä on pakollista, jotta järjestelmä voi kohdentaa palvelunsa jokaiselle käyttäjälle henkilökohtaisesti. Rekisteröitymisen yhteydessä käyttäjä voi halutessaan antaa myös tarkemmin muita henkilökohtaisia tietoja, kuten iän, sukupuolen ja ammatin, jotta personoinnista saataisiin vieläkin tarkempaa. Lisäksi käyttäjän sivuhistoriaa, syötteitä sekä valintoja seurataan ja tallennetaan. PNS-malli yhdistää siis sekä implisiittistä että eksplisiittistä toteutusta.

3. Esimerkkejä uutisten personoinnista

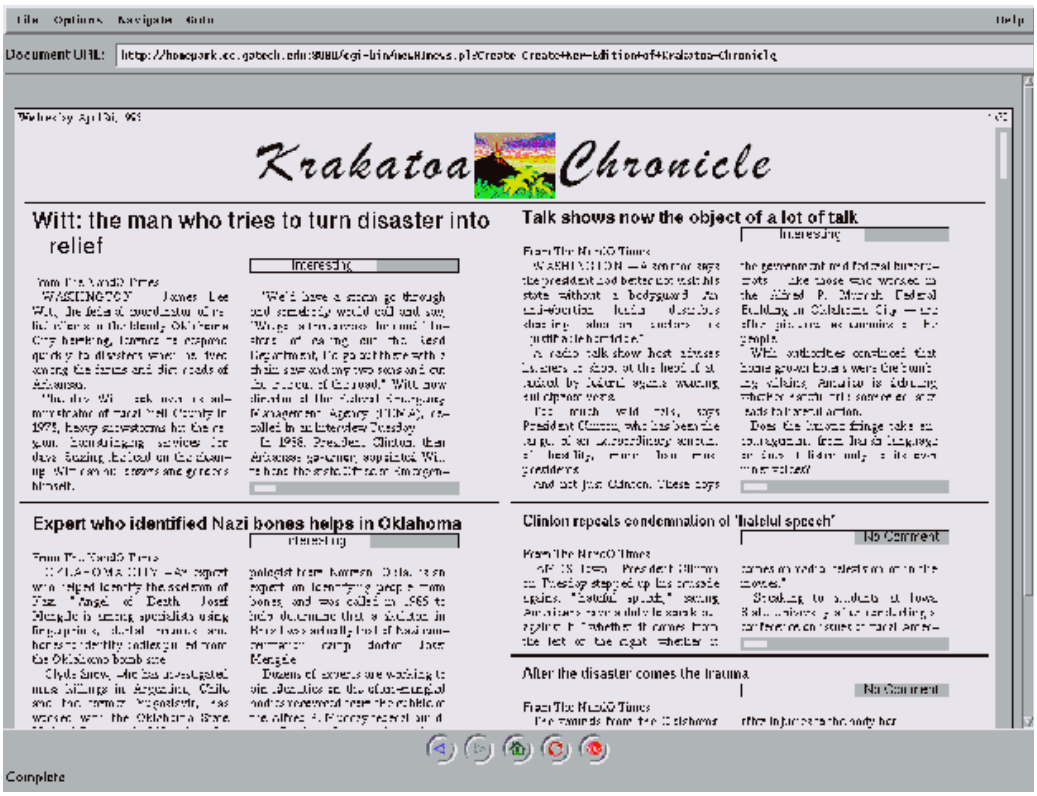
Useat sivustot ovat päätyneet hyödyntämään sekä eksplisiittistä että implisiittistä personointia. Tässä luvussa esittelen kaksi personoinnin mahdollistavaa uutissivustoa, jotka molemmat hyödyntävät kumpaakin edellä mainittua personoinnin tapaa.

3.1. Tapaus Krakatoa Chronicle

Uutisten personointi ei ole aivan viime vuosien ilmiö: jo vuonna 1995 otettiin ensimmäisiä askelia kohti henkilökohtaista Internetissä ilmestyvää sanomalehteä. Näistä ensimmäinen oli Krakatoa Chronicle, joka dynaamisesti mukautui käyttäjän tekemien valintojen mukaiseksi. Tämä oli ensimmäisiä Java-ohjelmointikieltä käyttäviä verkkosovelluksia, joka mahdollisti monimutkaisten rakenteiden hyödyntämisen. Krakatoa Chronicle mukailee kuvassa 1 esiteltyä uutisten personoinnin PNS-mallia. Tässäkin uutiset kerätään useista lähteistä.

Krakatoa Chronicle hyödyntää sekä implisiittistä että eksplisiittistä personointia. Eksplisiittiseen personointiin kuuluu käyttäjän mahdollisuus mukauttaa sivuston ulkoasua. Käyttäjä voi esimerkiksi valita, kuinka monta artikkelia ruudulla näkyy kerrallaan. Lisäksi jokaisen artikkelin lopussa on mahdollisuus pisteyttämiseen. Käyttäjä voi siis antaa haluamansa pistemäärän lukemalleen artikkelille riippuen siitä, kuinka mielenkiintoiseksi hän sen koki. Tämä

on nähtävissä kuvassa 3. Implisiittiseen personointiin puolestaan kuuluu käyttäjän toimintojen seuraaminen Java-agentin toimesta. Seurattuihin toimintoihin kuuluvat muun muassa käytetty aika ja vuorovaikutustekniikat (esimerkiksi vieritys, koon muuttaminen tai kurkistaminen). Näiden perusteella muodostuu henkilökohtainen käyttäjäprofiili. Käyttäjäprofiilia pystyy kuitenkin muuttamaan myös eksplisiittisesti avainsanojen avulla. [Kamba et al., 1995.]



Kuva 3: Krakatoa Chroniclen ulkoasu [Kamba et al., 1995]

Vielä vuonna 1995 pyrittiin mahdollisimman paljon konkreettista sanomalehteä muistuttavaan ulkoasuun. Kuvassa 3 on nähtävissä Krakatoa Chroniclen ulkoasu. Siinä on hyödynnetty useita palstoja, kuten paperisessakin sanomalehdessä. Uudempiin sivustoihin verrattuna tässä on muutamia mielenkiintoisia eroavaisuuksia. Kamba ja muut [1995] listaavat tärkeimpiin ulkoasuun liittyviin periaatteisiin seuraavat tekijät:

- Artikkeleiden tulisi olla nähtävillä kokonaisuudessaan.
- Otsikon tulisi mieluiten mahtua yhdelle riville.
- Suorakaiteen muotoinen tesselaatio on riittävä.
- Kuvia voidaan pienentää, jotta ne mahtuisivat haluttuun tilaan, mutta ne linkitetään alkuperäisversioon.

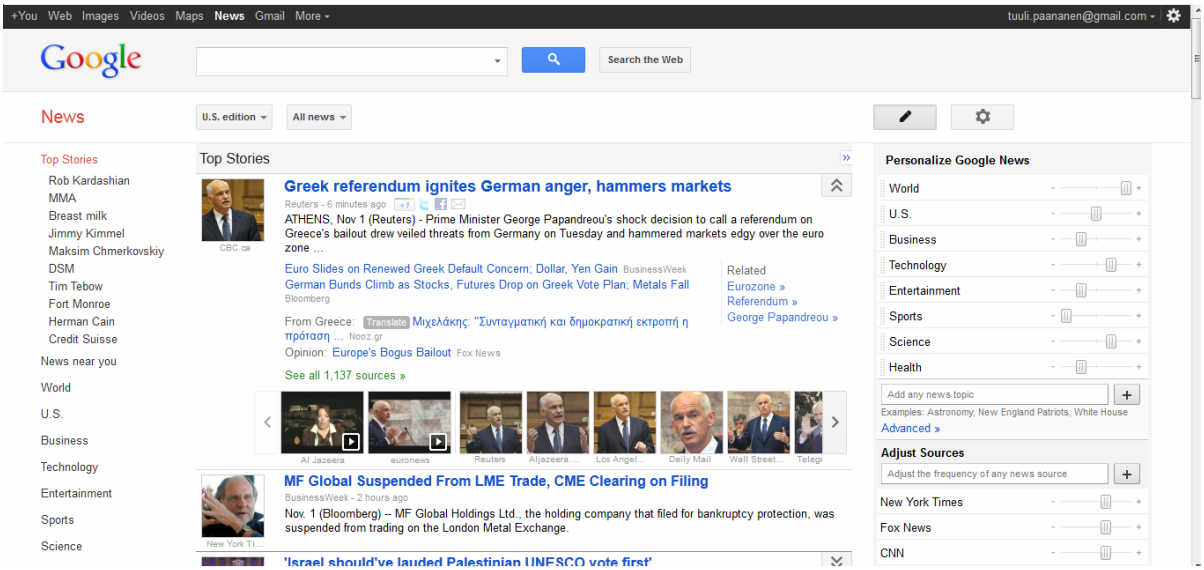
Näistä erityisesti ensimmäinen kohta eroaa nykykäytännöstä. Nykyisin lähes poikkeuksetta uutissivustoilla on etusivulla nähtävissä uutisotsikot, joiden alla on mahdollisesti kuva tai pieni tekstinpätkä. Joissakin tapauksissa nähtävissä on ainoastaan uutisotsikko. Tämä periaate poikkeaa paperisesta sanomalehdestä, joka on ollut Krakatoa Chroniclen esikuva.

3.2. Tapaus Google News

Modernimpi ja edelleen käytössä oleva uutisten personoinnin mahdollistava uutissivusto on Google News [Google News]. Sivustolla hyödynnetään muun muassa Liun ja muiden [2010] esittelemää käyttäjän klikkauskäytäntöön perustuvaa uutisten personointia. Google News on tietokoneella generoitu verkkosivu, joka kokoaa uutisotsikoita kansainvälisiltä uutispalveluilta. Sivustolla uutiset on jaoteltu taulukossa 2 esitellyn kategorisoinnin mukaisesti. Jotta käyttäjä pääsisi hyödyntämään palvelun tarjoamaa uutisten personointia, pitää hänen ensin kirjautua palveluun ja sen jälkeen vielä erikseen sallia sivuhistorian tallentaminen, jolloin järjestelmä pystyy seuraamaan hänen klikkauskäytäntöään [Liu et al., 2010]. Tämä onnistuu tavallisen Google-tilin avulla.

Google News hyödyntää taulukossa 2 esiteltyä yhteistoiminnallisen suodatuksen tekniikkaa. Järjestelmä muodostaa sekä implisiittisiä että eksplisiittisiä syötteitä hyödyntäen kullekin käyttäjälle henkilökohtaisen käyttäjäprofiilin. Käyttäjäprofiili muodostuu niiden kategorioiden avulla, joihin luetut artikkelit kuuluvat. Järjestelmä pyrkii implisiittisesti ennustamaan käyttäjää kiinnostavat artikkelit kategorioiden perusteella, jolloin näitä artikkeleita suositellaan käyttäjälle ensimmäisenä. Lisäksi käyttäjä voi eksplisiittisesti itse valita kiinnostuksen kohteensa. [Lavie et al., 2009.]

Kuvassa 4 on nähtävissä Google Newsin etusivu. Oikeassa reunassa on personointityökalu, jonka avulla käyttäjä voi valita, kuinka paljon hän haluaa lukea tiettyihin kategorioihin kuuluvia uutisia. Vaikka säätö kuvassa näyttää portaattomalta, on se todellisuudessa viisiportainen. Vaihtoehdot ovat *harvoin*, *silloin tällöin*, *joskus*, *usein* ja *aina*. Käyttäjä voi myös poistaa haluamansa kategorian kokonaan painamalla kategorian viereen ilmestyvää roskakoria kursorin ollessa kohdalla. Kuvassa näkyvien kahdeksan kategorian lisäksi käyttäjä voi lisätä haluamiaan kategorioita, jolloin ne ilmestyvät jo olemassa olevien kategorioiden alapuolelle. Tarjolla on myös mahdollisuus tarkempaan personointiin Advanced-painikkeen kautta. Sieltä käyttäjä näkee kaikki mahdolliset uutiskategoriat ja voi lisätä muita rajaavia tekijöitä, kuten rajauksen maantieteellisen sijainnin mukaan.



Kuva 4: Google Newsin eksplisiittinen personointi ja etusivu [Google News]

Google News eroaa paljon huomattavasti vanhemmasta verrokistaan Krakatoa Chroniclesta. Ulkoasulta ei ole haettu yhteneväisyyttä paperisen sanomalehden kanssa, vaan käyttäjälle on pyritty tekemään nopeasti silmäiltävissä oleva sivusto. Artikkeleita ei ole etusivulla kokonaan, vaan otsikkoa klikkaamalla pääsee alkuperäisartikkeliin. Google News kerää Krakatoa Chroniclen tapaan uutiset usealta sivustolta, jolloin artikkelia klikkaamalla päätyy jollekin muulle sivustolle, kuten esimerkiksi New York Timesin tai CNN:n sivustolle. Google News mahdollistaa jonkin asteisen ulkoasun mukauttamisen. Käyttäjä voi esimerkiksi valita, haluaako hän näkyville yhden vai kaksi palstaa.

4. Uutisten personointi käyttäjäkokemuksen näkökulmasta

Käyttäjäkokemus tarkoittaa käyttäjän tyytyväisyyttä ja uskollisuutta tiettyä sovellusta tai tuotetta kohtaan. Tämä saavutetaan sovelluksen tai tuotteen hyödyllisyydellä, helppokäyttöisyydellä sekä tuotteen käyttömukavuudella. [Kujala et al., 2011.] Uutisten personointiin sovellettuna tarkoitus on auttaa lukijoita löytämään heidän henkilökohtaisia kiinnostuksen kohteitaan vastaavia uutisia ja siten vähentää informaatioähkystä seuraavia negatiivisia vaikutuksia [Lavie et al., 2010]. Lisäksi sovelluksen tulisi olla käyttäjän kannalta niin hyödyllinen ja helppokäyttöinen, että hänen tekee mieli käyttää sitä jatkossakin. Personoinnin lähtökohtana on siis käyttäjä, mutta silti uutisten personointia on yleensä arvioitu järjestelmän näkökulmasta ja arvioijina ovat toimineet aiheeseen perehtyneet ekspertit. Koska järjestelmä on kuitenkin tarkoitettu tavallisille käyttäjille, pitäisi sitä myös arvioida heidän näkökulmastaan. [Diaz et al., 2008.]

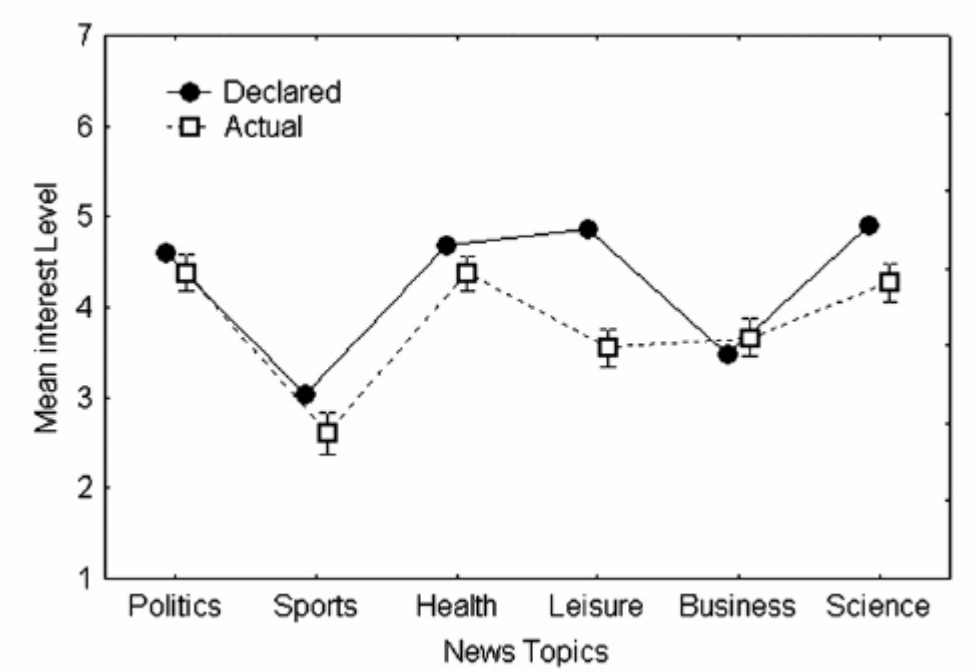
Uutisten personoinnin toteutuksessa on käyttäjäkokemuksen kannalta tiettyjä haasteita, jotka pitäisi pystyä huomioimaan suunnittelussa. Seuraavaksi esittelen kolme keskeisintä haastetta.

Ristiriitaiset lukemisen tavoitteet. Käyttäjät lukevat uutisia useista eri syistä, jotka voivat olla myös ristiriidassa keskenään. Uutisten lukemisen tavoitteena voi olla yleinen tiedonhankinta tai jonkin tietyn informaation etsiminen. Käyttäjät lukevat uutisia myös hedonistisista lähtökohdista, osana heidän vapaa-ajan viettoaan. Personoidut uutissivustot tukevat yleensä vain yhtä näistä lukemisen tavoitteista kerrallaan, sillä ne ovat usein ristiriidassa keskenään. Järjestelmä saattaa tarjota käyttäjälle vain sellaisia artikkeleita, jotka sopivat hänen käyttäjäprofiilissaan määriteltyihin kiinnostuksen kohteisiin, vaikka todellisuudessa käyttäjä haluaisi selailla artikkeleita vapaa-ajanviettotarkoituksessa ilman selkeää päämäärää. On siis vaikeaa personoida uutissivusto vastaamaan kaikkia käyttäjän tarpeita, joista erityisesti hedonistinen lukemisen tavoite on vaikein tyydyttää. [Lavie et al., 2010.] Hedonistinen lukemisen tavoite on kuitenkin tärkeää huomioida, sillä se on yksi merkittävimmistä tekijöistä hyvän käyttäjäkokemuksen syntymisessä [Kujala et al., 2011].

Sisällön suodattaminen käyttäjän kiinnostuksen kohteiden mukaiseksi. Uutisten personoinnin taustaoletuksena on, että käyttäjä osaa selvästi määrittää ja ilmaista kiinnostuksensa kutakin uutiskategoriaa kohtaan. Tämä oletus osoittautuu kuitenkin ongelmalliseksi, sillä Lavien ja muiden [2010] mukaan käyttäjien mielenkiinnon kohteet voivat olla monimutkaisia, ne voivat vaihtua ajan myötä, eivätkä ne aina ole tiedostettuja. Esimerkiksi käyttäjä, joka ei käyttäjäprofiilinsa perusteella ole kiinnostunut urheilusta, saattaa yllättäen haluta lukea jostakin tietystä urheilutapahtumasta (esimerkiksi jääkiekon maailmanmestaruuskilpailusta). Käyttäjä ei voi myöskään tietää olevansa kiinnostunut jostakin tietystä uutisesta ennen kuin hän on saanut sen luettavakseen. Kuten kuvasta 5 ilmenee, ongelmana on myös tutkimuksessa [Lavie et al., 2010] esille tullut kuilu käyttäjän todellisten ja ilmoitettujen kiinnostuksen kohteiden välillä. Kuvasta 5 voidaan havaita, että kuilu on kapeampi tiettyjen uutisaiheiden kohdalla. Politiikan osalta käyttäjän ilmoittama kiinnostus vastaa hyvin todellista tilannetta. Kauimpana toisistaan todellisuus ja käyttäjän ilmoittama mielipide ovat vapaa-ajan uutisten osalta. Tästä voidaan päätellä, että käyttäjät eivät aina osaa tarkalleen ilmaista mielenkiintoaan tiettyjä uutisaiheita kohtaan.

Sisällön suodattaminen käyttäjän kiinnostuksen kohteiden mukaiseksi on haastavaa myös siksi, että käyttäjän mielenkiinnon kohteet vaihtuvat ajan myötä. Tämän vuoksi on tärkeää ottaa huomioon käyttäjän lyhytkestoiset ja pitkäkestoisen kiinnostuksen kohteet. Lavien ja muiden [2010] mukaan personoidut uutissivustot keskittyvät vain niihin aiheisiin, joista käyttäjä oli kiinnostunut aiemmin, jolloin sivusto alkaa lopulta vaikuttaa tylsältä ja mielenkiinnottomalta. Käyttäjäkokemuksen kannalta juurikin käyttäjän pitkäkestoinen kiinnostus sivustoa kohtaan on tärkeää. Personoidun uutissivuston pitäisi siis pystyä mukautumaan käyttäjän mahdollisesti muuttuviin tarpeisiin ja mielenkiinnon koh-

teisiin, jotta hänelle muodostuisi mahdollisimman hyvä ja pitkäkestoinen käyttäjäkokemus. [Kujala et al., 2011.]



Kuva 5: Keskiarvot ja keskihajonnat ilmoitetusta ja todellisesta kiinnostuksesta tiettyjä uutisaiheita kohtaan. [Lavie et al., 2010]

Personoinnin syvyys. Jotta uutisten personointi olisi mahdollista toteuttaa käyttäjän kannalta mahdollisimman toimivasti, tulisi personoinnin oikea syvyys pystyä määrittämään. Hyvin pintapuolinen personointi nojautuu ainoastaan yleisiin uutiskategorioihin, joita ovat esimerkiksi urheilu ja talous. Käyttäjä saattaa valita urheilu-uutiset kiinnostuksen kohteekseen ja näin ollen saada kaikki urheilu-uutiset luettavakseen, vaikka oikeasti hän olisi kiinnostunut vain jostakin tietystä urheilulajista. Pintapuolisesta personoinnista on siis seurauksena liian laaja uutisvirta. Jos taas personoinnissa mennään liian syvälle, eli keskitytään avainsanoja käyttäen esimerkiksi vain johonkin tiettyyn urheilujoukkueeseen, voi käyttäjän saama tiedonmäärä yksipuolistua ja vähentyä, eikä näin ollen vastaa kaikkia käyttäjän tarpeita. Diazin ja muiden [2008] mukaan käyttäjät kuitenkin pitävät enemmän avainsanojen avulla tehtävästä syvällisemmästä personoinnista, koska silloin he voivat määritellä kiinnostuksen kohteensa hyvin tarkasti verrattuna kategorioiden avulla tehtävään pintapuolisempaan personointiin.

5. Uutisten personoinnin vaikutukset tulevaisuuden journalismille

5.1. Portinvartijateoria

Tiedotusopillisessa ajattelussa journalismi on totuttu näkemään ikään kuin yhteiskunnan portinvartijana, joka valikoi laajasta tietovirrasta ne asiat, jotka pääsevät yleisön tietoisuuteen, toisin sanoen ylittävät uutiskynnyksen [Nordenstreng, 1978]. Uutisten personointi siirtää portinvartijan roolin toimittajilta uutisten lukijoille, jolloin on aiheellista pohtia, yksipuolistuuko ihmisten saama tieto. Kun personoidun verkkolehden lukija saa eteensä vain niitä uutisia, joista hän henkilökohtaisesti on kiinnostunut, on vaarana maailmankuvan kapeneminen.

Huoli maailmankuvan kapenemisesta on mielestäni suurempi ongelma nimenomaan käytettäessä implisiittistä uutisten personointia. Implisiittistä personointia käyttävä tietokoneohjelma seuraa käyttäjän lukemia uutisia ja niiden aihealueita, eikä tämä vaadi käyttäjältä minkäänlaisia toimenpiteitä palveluun rekisteröitymistä lukuun ottamatta. Tällöin uutistarjonnan mukautuminen käyttäjän mielenkiintojen mukaiseksi ei vaadi tältä minkäänlaisia aktiivisia valintoja, eikä hän välttämättä myöskään tiedosta saamansa tiedon yksipuolistuneen. Eksplisiittinen personointi toimii käyttäjän aktiivisten valintojen perusteella, jolloin käyttäjä pystyy paremmin näkemään tekemänsä valinnat ja halutessaan myös muuttamaan niitä myöhemmin. Tämäkin voi toki johtaa maailmankuvan vääristymiseen, mutta tässä tapauksessa uutisten mukautumisperiaate on läpinäkyvämpi.

Thurmanin [2011] mukaan journalistit sekä puolustavat että vastustavat uutisten personointia. Suurin osa osoitti huolensa yllätyksellisyyden (serendipity) katoamisesta: kun personointi viedään tarpeeksi pitkälle, ei lukija voi yllättyä positiivisesti löytäessään ennalta arvaamattoman mielenkiintoisen artikkelin. Vain yksi tutkimukseen osallistuneista journalisteista esitti vastalauseen tätä huolenaihetta vastaan. Hänen mukaansa personoinnin voi toteuttaa siten, että se sallii yllätyksellisyyden.

Toinen huolenaihe osuu lähelle maailmankuvan vääristymisen riskiä; toimittajat epäilivät, osaavatko lukijat todella valita itse, millaisista uutisista he ovat kiinnostuneita. Thurman [2011] kuitenkin painottaa, että käyttäjän personoinnin myötä saamaa toimituksellista roolia ei ole tutkittu tarpeeksi, jotta siitä voitaisiin tehdä kovin pitkälle meneviä johtopäätöksiä.

5.2. Journalistin työ

Seuraavaksi arvioin uutisten personoinnin vaikutusta journalistin työhön. Aitamurto [2009] on tutkinut ammattijournalismin kohtaloa personoitujen uutisten aikakaudella erityisesti Yhdysvalloissa, jossa paperiset sanomalehdet ovat jo

pitkään menettäneet suosiotaan. Tämä näkyy sekä journalistien että uutissisältöjen vähenemisessä. Verkkolehtien ja erityisesti personoinnin kautta journalismin olisi kuitenkin mahdollista tarjota lukijalle jotakin paperista sanomalehteä parempaa. Tämänhetkistä tilannetta Aitamurto [2009] arvioi seuraavasti: ”Verkkojulkaisut antavat esimerkiksi lukijan personoida sisältönäkymää vain vähän, jos ollenkaan. Interaktiivisuutta on vähän: kommentteja ei voi yleensä arvostella, eikä juttuja äänestää ylös tai alas suosittujen listassa”. Tässä personointi nähdään siis ennemminkin journalismin pelastajana kuin sen tuhoajana.

Toinen tapa lähestyä personoinnin vaikutusta journalistin työhön on taloudellinen. Uutisyhtiön on helpompi kohdentaa mainontaa, kun sillä on tiedossa, minkälaisista asioista kukin lukija pitää. Näin mainostulot saadaan kasvuun. Kun tietokonealgoritmi valitsee kiinnostavat ja mainonnan kannalta suotuisat uutiset sivustolle, ei siihen työhön tarvita kallista toimittajaa. [Thurman, 2011.] Tästä seuraa se, että pahimmassa tapauksessa uutistoimitukselle käy samalla tavalla kuin suurelle osalle teollisuudesta: koneet korvaavat ihmisen.

Aitamurto [2009] esittää kuitenkin lohdullisemman näkymän: toimittajille syntyy uusia tehtäviä nimenomaan tietotekniikan ja Internetin myötä. Hänen mukaansa freelance-toimittajat lisääntyvät, mikä puolestaan tarkoittaa sitä, että journalismista tulee useille enemmänkin tehtävä kuin ammatti. Toinen kanava on blogit, joita hyödyntävät erityisesti aloittelevat journalistit. Lisäksi Aitamurto esittelee aivan uudenlaisen termin tulevaisuuden journalisteille: *journalpreneur*, journalistin ja yrittäjän yhdistelmä. Näillä journalpreneureilla on loistavat teknologiset taidot sekä usein jonkinlaista yrittäjyystaustaa. Vastaavanlaisia journalpreneureja löytyy jo Yhdysvalloista, jossa he voivat vapaasti kokeilla uusia toimituksellisia ilmiöitä verkkolehtiensä kautta.

6. Yhteenveto

Tässä tutkielmassa olen esitellyt uutisten personointia. Ensin kävin läpi uutisten personoinnin toteutusta jakaen sen eksplisiittiseen ja implisiittiseen personointiin. Eksplisiittinen toteutus on läpinäkyvämpi vaihtoehto, koska siinä käyttäjä pääsee itse tekemään valinnat henkilökohtaisen verkkolehden toteuttamiseksi. Monet tutkijat ovat kuitenkin sitä mieltä, ettei käyttäjillä ole mielenkiintoa tai aikaa näiden monimutkaisten valintojen tekemiseen. Lisäksi on syytä pohtia, osaavatko käyttäjät itse määritellä, millaisista uutisista he oikeasti ovat kiinnostuneita. Implisiittinen personointi puolestaan tapahtuu tietokoneella generoitujen algoritmien avulla ja kaikki toiminta tapahtuu käyttäjän ulottumattomissa. Tehokkaimmaksi tavaksi nouseekin eksplisiittisen ja implisiittisen personoinnin yhdistelmä, jossa käyttäjä pääsee tekemään joitakin valintoja samaan aikaan, kun tietokonealgoritmi seuraa käyttäjän toimintoja verkkosivulla. Yhdessä näistä muodostuu käyttäjäprofiili.

Uutisten personoinnin esittelyn jälkeen kävin läpi kaksi esimerkkiä personointia hyödyntävistä uutissivustoista: Krakatoa Chroniclen ja Google Newsin. Vuodelta 1995 olevaa Krakatoa Chroniclea voidaan pitää ensimmäisenä uutisten personointia hyödyntäneenä verkkosivuna. Ulkoasuun ja toimitoihin liittyvät seikat ovat vuosien saatossa muuttuneet, mutta pääperiaatteiltaan Krakatoa Chronicle ja modernimpi Google News ovat hyvin lähellä toisiaan. Kummankin toimintaperiaatteet mukailevat luvussa 2 esiteltyä uutisten personoinnin PNS-mallia [Paliouras et al., 2006].

Neljännessä luvussa pohdin uutisten personointia käyttäjäkokemuksen kautta. Uutisten personointi on suunniteltu helpottamaan uutisten lukemista Internetin aikakaudella, jolloin kiinnostavan informaation löytäminen valtavasta tietovirrasta voi olla aikaa vievää. Sivustoja arvioidaan kuitenkin usein järjestelmän näkökulmasta, jolloin käyttäjä voi jäädä vähemmälle huomiolle. Listasin muutamia haasteita, joita uutisten personointiin liittyy käyttäjäkokemuksen kannalta. Nykyiset järjestelmät eivät vielä ole pystyneet vastaamaan kaikkiin näihin haasteisiin, joten tarvitaan lisää käyttäjätutkimusta.

Viidennessä luvussa esittelin uutisten personointiin liittyviä arvokysymyksiä. Portinvartijateoria kyseenalaistaa journalistien roolin personoitujen uutisten aikakaudella. Tutkijat ja toimittajat ovat huolissaan lukijoiden maailmankuvan kapenemisesta, jos nämä saavat luettavakseen ainoastaan artikkeleja, joista he ovat kiinnostuneita. Tällöin katoaa yllätyksellisyys ja uuden löytämisen ilo. Tämä johtaa väistämättä siihen, että journalistien työnkuva on muutoksen tilassa. Journalistit joutuvat jakamaan portinvartijan roolin lukijoiden kanssa algoritmien hoitaessa uutisten suodatuksen heidän puolestaan [Thurman, 2011].

Uutisten personointi ei varsinaisesti ole uusi asia. Ihmiset ovat sanomalehden alkuajoista lähtien ”suodattaneet” vähemmän mielenkiintoiset uutiset ja lukeneet vain mielenkiintoisimmat. Teknologia on jo sillä tasolla, että on mahdollista toteuttaa hyvin tarkasti käyttäjän mielenkiinnon kohteet tunnistava uutissivusto. Lisätutkimusta kaipaakin mielestäni käytettävyys- ja käyttäjätyytyväisyyspuoli. Useammassa kohdassa toin esille, kuinka käyttäjät ovat haluttomia ottamaan erityisesti eksplisiittiset personointitavat käyttöönsä. Personointi pitäisi tehdä käyttäjille helpoksi ja miellyttäväksi kokemukseksi, jota lähelle implisiittisellä personoinnilla on jo päästykin. Lisäksi on mielenkiintoista seurata, kuinka uutisten personointiin liittyvä arvokeskustelu etenee, ja muuttuuko toimittajien työ todella Aitamurron [2009] kuvailemaksi *journalpreneurismiksi*. Yksi on kuitenkin varmaa: personoidut verkkouutiset tuskin vievät tilaa paperiselta sanomalehdeltä, sillä sanomalehti kuuluu edelleen monen aamiaispöytään yhtä oleellisesti kuin aamukahvi.

Viiteluettelo

[Aitamurto, 2009] Tanja Aitamurto, *Kymmenen väitettä journalismin tuhosta – ja miksi niistä ei kannata huolestua*, Raportti journalismin trendeistä Yhdysvalloissa vuonna 2009. Helsingin Sanomain Säätiö, 2009.

[Díaz et al., 2008] Alberto Díaz, Antonio García and Pablo Gervás, User-centered versus system-centered evaluation of a personalization system. *Information Processing and Management* **44** (2008), 1293-1307.

[Google News] Google News, http://news.google.com.

[Kamba et al., 1995] Tomonari Kamba, Krishna Bharat and Michael C. Albers, The Krakatoa Chronicle - an interactive, personalized newspaper on the Web. In: *Proceedings of the 4th International World Wide Web Conference* (1995), 11-14.

[Kujala et al., 2011] Sari Kujala, Virpi Roto, Kaisa Väänänen-Vainio-Mattila, Evangelos Karapanos and Arto Sinnelä, UX-curve: a method for evaluating long-term user experience. *Interacting with Computers* **23** (2011), 473-483.

[Lavie et al., 2010] Talia Lavie, Michal Sela, Ilit Oppenheim, Ohad Inbar and Joachim Meyer, User attitudes towards news content personalization. *International Journal of Human-Computer Studies* **68** (2010), 483-495.

[Liu et al., 2010] Jiahui Liu, Peter Dolan and Elin Rønby Pedersen, Personalized news recommendation based on click behavior. In: *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI '10)*, (2010), ACM, 31-40.

[Nordenstreng, 1978] Kaarle Nordenstreng, *Tiedotusoppi: Johdatus yhteiskunnallisten viestintäprosessien tutkimukseen*. Helsinki, WSOY, 1978.

[Paliouras et al., 2006] Georgios Paliouras, Mouzakidis Alexandros, Christos Ntoutsis, Angelos Alexopoulos and Christos Skourlas, PNS: Personalized multi-source news delivery. In*: Proceedings of the 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems (KES)*, Springer, 1152-1161.

[SVT, 2009] Suomen virallinen tilasto (SVT): Joukkoviestimet (verkkojulkaisu), 2009. Helsinki: Tilastokeskus. Saatavana: http://www.stat.fi/til/jvie/2009/jvie_2009_2010-12-10_tie_002.html

[SVT, 2010] Suomen virallinen tilasto (SVT): Joukkoviestimet (verkkojulkaisu), 2010. Helsinki: Tilastokeskus. Saatavana: http://www.stat.fi/til/jvie/2010/jvie_2010_2011-05-26_tie_001_fi.html

[Thurman, 2011] Neil Thurman, Making 'The Daily Me': Technology, economics and habit in the mainstream assimilation of personalized news. *Journalism* **12**, 4 (May 2011), 395-415.

Internet-pohjaisen verkon valvontamenetelmät

Janne Redsven

Tiivistelmä.
Tämä tutkielma käsittelee Internet Protocol -pohjaisen verkon seurantaa. Seuranta perustuu havaintotietojen keruuseen ja hyödyntämiseen tarkoituksenmukaisella tavalla. Tutkielma pyrkii luomaan esimerkein käsityksen siitä, miksi seurantaa tulisi tehdä ja mitä hyötyjä siitä on.

Tutkielmassa luodaan katsaus keskeisiin havaintotiedon keruuta tukeviin protokolliin ja pohditaan niiden sopivuutta käyttötarkoitukseensa sekä niiden hyödyntämiseen liittyviä kysymyksiä. Seurantajärjestelmien suunnitteluun ja käyttöön liittyy myös oppiminen, jonka tukeminen tulisi olla keskeinen peruste suunnittelussa.

Tekniikan kehittyessä myös erilaiset seurannan menetelmät ja mekanismit kehittyvät. Siksi tutkielman loppuosassa pohditaan vielä sitä, mitä kaikkea tutkielmassa esitellyillä protokollilla ei voida seurata.

Avainsanat ja -sanonnat: Tietoverkko, seuranta.
CR-luokat: C.2.2, C.2.3

1. Johdanto

Yhä useampi päivittäin käytetty palvelu on riippuvainen tietoverkosta joko suoraan tai välillisesti. Paikallisesti asennettu palvelu tai sovellus saattaa toiminnassaan hyödyntää esimerkiksi tietokantaa, verkkolevyä tai muita tietoverkon palveluja kuten verkkoon liitettyä tulostinta. Tällöin ongelmat tietoverkon osissa saattavat näyttäytyä yllättävänä oireiluna käyttäjälle. Helppokäyttöisinkin sovellus menettää käytettävyytensä myötä käyttäjänsä, mikäli se ei toimi.

Tietoverkon tilan jatkuva seuranta saattaa mieltyä luonteeltaan toiminnoksi, jonka välttämättömyys ei ole itsestään selvää. Ikävä kyllä seurannan vahvuus — tai sen puute — näyttäytyykin vasta silloin, kun seurannan avulla saatavalle tiedolle olisi käyttöä. Seurannan voi kuitenkin ajatella tähtäävän omalta osaltaan sujuvan käyttäjäkokemuksen ylläpitoon eli siihen, että ongelmiin on reagoitu jo ennen kuin ne näkyvät käyttäjille.

Tämän tutkielman tarkoituksena on esitellä joitakin keinoja tietoverkon perustoimintojen seurantaan. Seuranta-sanan tilalla voisi hyvin käyttää sanaa valvonta, joka ehkä kuvaisi paremmin itse toimintaa. Molemmilla sanoilla on nyky-yhteiskunnassa hieman orwellilainen, negatiivinen merkitys. Tämän tutkielman yksi tarkoitus on kuitenkin osoittaa, että tarkoituksenmukaisin keinoin

toteutetulla valvonnalla pyritään takaamaan käyttäjälle tietoverkon palvelujen saatavuus. Samalla pyritään turvaamaan käyttäjän yksityisyyden suoja.

Tutkielmassa käytetään kaikelle kerätylle tiedolle nimitystä havaintotieto. Nimityksellä pyritään neutraaliuteen, sillä kerätty tieto itsessään on laadutonta ja vasta sen hyödyntäminen seuranta- tai valvontatarkoituksessa muuttaa havaintotiedon seurantatiedoksi. Yhtä hyvin saman tiedon pohjalta voitaisiin luoda anonyymiä tilastotietoa. Kyse on siis siitä, miten ja millaiseen tarkoitukseen tietoa hyödynnetään.

Vaikka tutkielman näkökulma on periaatteessa käyttäjälähtöinen, siinä ei kuitenkaan tarkastella käyttäjää koskevia seikkoja, kuten yksityisyyden säilymistä esimerkiksi Sähköisen viestinnän tietosuojalain tarkoittamien tunnistamistietojen osalta. Toisaalta tietoisuus tietoverkon tilan seurantaan käytettävistä menetelmistä ja käsitys niiden tehtävästä voi auttaa ymmärtämään, miksi verkkoliikennettä tulee valvoa.

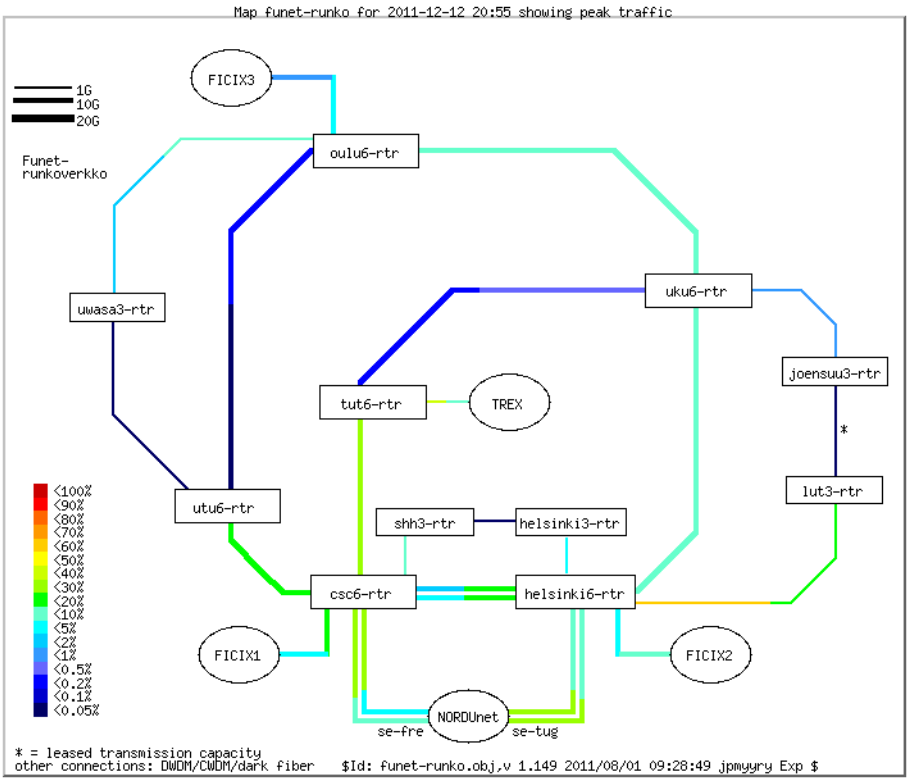
Tämä tutkielma pyrkii, aihepiiristään huolimatta, välttämään ylimääräistä teknisyyttä. Tutkielma rajautuu käsittelemään Internet- tai lähiverkon seurantaan käytettäviä työkaluja. Siksi lukijan kuitenkin oletetaan jossakin määrin hallitsevan Internet Protocol (IP) -verkon toiminnan perusperiaatteita, vaikka niihin ei syvällisesti pureudutakaan.

2. Havaintotiedon käyttötarkoituksista

Eräs menetelmä tietojen keruun tarpeiden hahmotteluun voisi olla aiheen lähestyminen käyttötapakuvausten kautta (Shiravi, Shiravi & Ghorbani, 2011). Käyttötapakuvauksella tarkoitetaan listausta toiminnasta ja vuorovaikutuksesta käyttäjän tavoitteiden saavuttamiseksi. Käyttötapakuvauksien tarkoituksena tässä voisi olla rakentaa mahdollisia tilanteita, joissa kerättyä havaintotietoa tarvitaan. Voitaisiin kuvitella esimerkiksi ylläpitäjä, joka on kiinnostunut runkoverkon kuormituksesta ja haluaa yhdellä silmäyksellä nähdä kuormitustilanteen. Yhdessä verkkotopologiaa havainnollistavan kuvan kanssa voidaan nimetä yhteysvälejä ja verkkolaitteita, jotka ovat oleellisia ja tätä kautta seurata niiden kautta kulkevaa liikennettä.

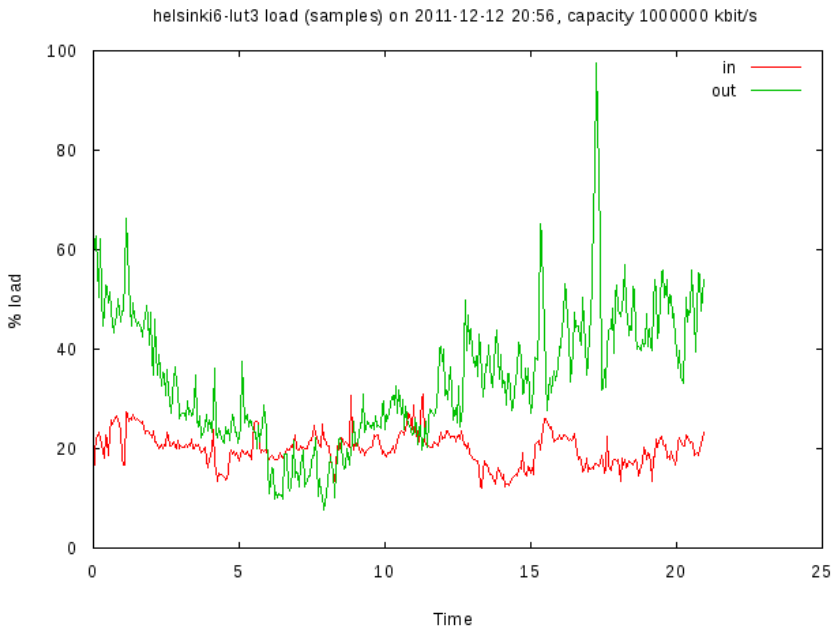
Eräs sovellutusesimerkki on Finnish University and Research Network -tietoverkon (Funet) ”sääkartta” (ks. kuva 1). Kuvasta nähdään esimerkiksi, että Helsingin (helsinki6-rtr) ja Lappeenrannan (lut3-rtr) välisen suoran yhteyden, jonka tiedonsiirtonopeus on 1 gigabittiä sekunnissa, hetkellinen kuormitus (peak traffic) on 50—60 %. Tämä tieto on olennainen, sillä jos käyttöaste lähentelee 100 %:a, esimerkiksi palvelunestohyökkäyksen tai muutoin suuren käyttöasteen vuoksi, tätä yhteysväliä käyttävä liikennöinti hidastuu tai voi lakata kokonaan. Yhteysväliä voi tutkia tarkemminkin: klikattaessa sääkartalla yh-

teysväliä helsinki6-rtr—lut3-rtr valvontajärjestelmän kautta on mahdollista tarkastella graafisesti esimerkiksi mainitun yhteysvälin prosentuaalista kuormitusta (ks. kuva 2). Tästä kuvasta nähdään muun muassa, että tarkasteluhetken 50—60 % kuormitus on ”ulospäin” helsinki6-rtr:iin nähden ja kuormitus ”sisään” on noin 20 %, eli Lappeenrannan suuntaan on enemmän liikennettä. Yhteysvälin kuormitus on myös kertaalleen ollut lähellä 100 %:a ja tämä voi, tilanteesta riippuen, antaa aihetta selvittää kuormituksen syitä.



Kuva 1. Funet-runkoverkon ”sääkartta” (CSC, 2011)

Vastaavalla tavalla voitaisiin tarkastella esimerkiksi Helsinki—Lappeenranta -yhteysvälin pakettimääriä (paketti on IP-protokollan perusyksikkö) tai yhteysvälillä ilmeneviä virheitä, kuten kehysvirheitä tai virheitä tämän yhteysvälin reitittimien porttitasolla. On kenties hieman pessimististä pohtia mahdollisia häiriötilanteita, mutta asiaa voidaan tarkastella myös siitä näkökulmasta, että tieto ongelmista on ylläpidon kannalta mukavampaa saada omista järjestelmistä mahdollisesti jo ennalta käyttäjien tai teknisen tuen yhteydenottojen sijaan. Tietoa voidaan haluta kerätä vain tilastolliseen käyttöön. Toisaalta vain tilastollisesta näkökulmasta kerätty informaatio saattaa olla luonteeltaan liian epätarkkaa, eikä sen vuoksi auta virhetilanteissa. Kuitenkin virhetilanteita varten kerätty tieto voidaan tuonnempana havaita merkitykselliseksi tilastolliseltakin kannalta.



Kuva 2. Liikennekuvaaja yhteysväliltä helsinki6-rtr—lut3-rtr (CSC, 2011)

Yhä merkityksellisempää lienee esimerkiksi Sähköisen viestinnän tietosuojalain (SVTSL) (16.6.2004/516, 19 §) esittämä vaade huolehtia siitä, että organisaation tietoverkko on turvallinen käyttää. Turvallisuudella tarkoitetaan sitä, että verkon tietoturvan tulee täyttää kolme vaatimusta: 1) luottamuksellisuus (confidentiality), eli tieto on vain valtuutettujen käyttäjien ulottuvilla, 2) eheys (integrity), eli tieto on vain valtuutettujen käyttäjien muokattavissa eikä tieto muutu hallitsemattomasti esimerkiksi häiriötilanteissa ja 3) saatavuus (availability), jolla tarkoitetaan sitä, että tieto on valtuutettujen käyttäjien saatavilla, kun he sitä tarvitsevat (Valtiovarainministeriö, 2010, 29). Tämä on tietenkin itsestään selvää organisaatioissa, joissa käsitellään esimerkiksi henkilö-, viranomais- tai muita salassa pidettäviä tietoja. Kuitenkin laajat ongelmat tietoturvallisuuden ylläpidossa poikivat vähintään kielteistä julkisuutta ja harva on mielissään minkään henkilökohtaisen tiedon joutumisesta vääriin käsiin.

Käyttäjät myös saattavat suhtautua tietoturvallisuuteen leväperäisesti, sillä oman käsityksensä mukaan heillä ei ole mitään ”varastamisen arvoista”. Viestintävirastossa toimiva kansallinen tietoturvaviranomainen CERT-FI kuitenkin varoittaa muun muassa siitä, että hämärin keinoin hankittuja sähköpostiosoitteita voidaan käyttää esimerkiksi roskapostin lähettämiseen väärentämällä lähettäjäosoitteeksi varastettu sähköpostiosoite tai hyödyntämällä käyttäjän tunnusta sähköpostipalvelimelle (CERT-FI, 2010a). CERT-FI listaa joukon tyypillisimmistä käyttäjältä varastettaviksi tiedoiksi muun muassa eri palvelujen kirjautumistiedot, luottokortti-, pankkitili- ja identiteettitiedot (CERT-FI, 2010b,

7—8). Tällä perusteella voidaan olettaa, että jokaisella käyttäjällä on vähintään ”verkko-minä”, identiteetti, jota voidaan helposti hyödyntää rikollisiin puuhiin.

Tässä tutkielmassa keskitytään lähinnä tarkastelemaan tietoliikenneverkon saatavuuteen liittyviä kysymyksiä. Yleisenä suunnitteluperusteena voisi kuitenkin pitää tietoturvallisuuden ylläpitoa, eli myös luottamuksellisuuden ja eheyden säilyttämistä, jonka eräs muoto on näissä ilmenevien poikkeamien tunnistaminen ja niihin reagointi. Esimerkiksi viruksille tai muille haittaohjelmille saattaa olla tunnistettavissa ominainen tapa liikennöidä verkossa ja tällöin historiatiedon analysoinnilla voidaan kenties tuonnempana tarkastella niiden levinneisyyttä, pyrkiä jäljittämään alkuperää ja tunnistaa saastuneet työasemat tai verkkolaitteet. Historiatiedolla voidaan tarvittaessa tehdä myös rikosoikeudellista tutkintaa. Vaikka virka-apupyynnöt Suomen ulkopuolelle tuottavatkin valitettavan vähän tulosta, mahdollisimman tarkat tiedot auttavat tapauksen selvittelyssä. (CERT-FI, 2010b, 22)

Tutkinnassa on myös toinen puoli, sillä organisaation tietoverkon käyttäjät voivat syyllistyä laittomuuksiin, joko tahallisesti tai tahattomasti. Niinpä suomalainen oikeuslaitos voi edellyttää tietojen luovuttamista nimetyistä käyttäjistä, jolloin asiallisesti toteutettu valvonta vähentää epäselvyyksiä ja vähintäänkin sujuvoittaa yhteistyötä eri toimijoiden välillä.

Oletettavaa on, että mitä aiemmin potentiaaliseen poikkeamaan reagoidaan, sitä paremmin ongelmat saadaan rajattua ja selvittelyyn kuluva kokonaistyöaika pienenee. Ja vaikka ylläpito ei ajattelisi vain itseään, sen aktiivisuutta voitaneen myös pitää vähintään kohteliaisuutena suojella toisia käyttäjiä esimerkiksi virusten aiheuttamilta harmeilta.

3. Miten havaintotietoa kerätään?

Propagandatutkija Harold Lasswell kiteytti 1940-luvulla viestinnän siirtomallin kaavassaan ”kuka sanoo mitä minkä kanavan välityksellä kenelle ja millä vaikutuksella” (ks. esim. Pietilä, Malmberg & Nordenstreng, 2004). Tämä viestinnän kaava sopinee eräänlaiseksi rungoksi myös tietoverkossa kulkevan liikenteen seurannalle: häiriötilanteessa on tärkeää tietää, mistä liikenne tulee (kuka sanoo), mitä reittiä tai yhteyttä käyttäen (minkä kanavan välityksellä), mihin liikenne menee (kenelle) ja mitä se aiheuttaa (millä vaikutuksella). Näihin kysymyksiin pyritään vastaamaan keräämällä havaintotietoa.

Havaintotietojen keruu voidaan ajatella kahden eri viestijän välillä tapahtuvana viestin vaihtona: asiakas (esimerkiksi seurantaohjelmisto) tiedustelee kohteelta (esimerkiksi verkkolaite tai palvelin) erilaisia ominaisuuksia tai attribuutteja. Viestintä voi tapahtua myös toiseen suuntaan eli verkkolaite tai palvelin voi lähettää viestejä seurantaohjelmistolle. Koska ominaisuuksia ja laitevalmis-

tajia voi olla mielivaltainen määrä, on ollut välttämätöntä sopia jonkinlaisista viestinnän säännöistä. Kutsun tässä tutkielmassa näitä sääntöjä yhteyskäytännöiksi eli protokolliksi.

Ersue ja Claise (2011) esittelevät IP-verkonvalvonnan protokollia, joita ovat Simple Network Management Protocol (SNMP), SYSLOG ja IP flow information export (IPFIX) sekä Packet sampling (PSAMP). Tämä jaottelu koskee ainoastaan Internet Protocol (IP) -pohjaisia verkkoja ja on luonnollisestikin vain eräs jaottelu. Uudempia ja toisiin käyttöaiheisiin soveltuvia tekniikoita on olemassa, ja näistä muun muassa Subramanian, Gonsalves ja Rani (2010, 96—98) ovat laatineet koosteen. Tutkielman tarkastelun kohteena on kuitenkin tavanomaisten IP-tietoverkkojen toiminnallisuuden seuranta ja menetelmien esittely, ja siksi Ersuen ja Claisen jaottelu sopii riittävän yksinkertaisuutensa vuoksi. Samalla se kuitenkin antaa hyvän pohjan eri seurantamekanismien ymmärtämiseksi yleisellä tasolla.

Seuraavissa kohdissa käydään läpi Ersuen ja Claisen (2011) esittelemät protokollat ja pohditaan samalla niiden mahdollisia käyttötarkoituksia ja toiminnallisia eroja.

3.1. Simple Network Management Protocol

Internet Engineering Task Force (IETF) julkaisi vuonna 1988 ensimmäisen version Simple Network Management Protocol -standardista (SNMP) (RFC 1067, 1988). Stallings (1998) esittelee SNMP-standardin kolme tärkeää ominaisuutta: 1) se määrittelee protokollan, jolla yksi tai useampi hallintajärjestelmä ja useat eri laitteet (agentit) voivat vaihtaa tietoa, 2) se määrittelee esitysmuodon välitettävälle ja laitteeseen tallennettavalle hallintatiedolle (management information) ja 3) määrittelee useita yleiskäyttöisiä objekteja (tai muuttujia), joihin tieto kustakin ominaisuudesta säilötään. Tällaisien objektien kokonaisuutta kutsutaan nimellä Management Information Base (MIB). MIB-kokonaisuus on määritelty standardissa, joten toiminnallisuudeltaan samankaltaisissa järjestelmissä voi olettaa olevan samoja (saman nimisiä) objekteja, joita voi käsitellä samalla tavoin. Toisin sanoen eri valmistajien laitteet, jotka tukevat standardia, antavat esimerkiksi tietoa tilastaan saman objektin (joka on käytännössä muuttuja) kautta.

Valmistajilla on myös mahdollisuus laajentaa objektien kokoelmaa kirjoittamalla omia laajennoksiaan (vendor extensions). Tällöin valmistaja voi tarjota lisäobjekteja, jotka tarjoavat esimerkiksi tarkempia tietoja laitteen tilasta verkon ylläpitäjien hyödynnettäväksi. (Stallings, 1998, 37)

Mainittakoon vielä, että SNMP-standardi mahdollistaa ominaisuuksien kyselyn lisäksi myös laitteiden tilan ja asetusten hallintaan liittyviä toimintoja:

asetuksia voi muuttaa, eli hallintajärjestelmä voi tiedustella esimerkiksi muuttujan arvoa ja tarpeen mukaan muuttaa sen toiseksi ja laite voi tarvittaessa lähettää tilanmuutosviestin (trap) määritellylle vastaanottajalle. Ylläpitäjä voi määritellä laitekohtaisia ehtoja tilaviestien lähetykseen merkittävistä verkkoongelmista. Tällaisia voisivat olla esimerkiksi laitteiden välisen verkkoyhteyden kuormituksen, joka ylittää tietyn raja-arvon, tai laitteiden välisten yhteyksien katkeaminen. Tämän yksinkertaisen toiminnallisuuden ansiosta toiminnan seuranta on kevyttä, eikä hallintajärjestelmän tarvitse tältä osin käydä säännöllisesti läpi jokaista hallittavaa laitetta. Toisaalta myös hallintajärjestelmä voidaan asettaa reagoimaan eri tavoin tilaviesteihin, joten hälytykset saadaan lähes reaaliaikaisesti eteenpäin vaikkapa tekstiviestinä.

SNMP:n ensimmäinen versio, sittemmin kirjoitusasultaan SNMPv1, saavutti nopeasti suosiota ja laitevalmistajat alkoivat noudattaa sitä järjestelmissään. Ikävä kyllä SNMPv1:n kehityksessä ei huomioitu isojen tietomäärien kyselyä yhdellä kertaa ja sen tietoturvallisuus oli puutteellista. Muun muassa näitä ongelmia ratkomaan kehitettiin SNMPv2 (RFC 1441), joka standardoitiin vuonna 1993. (Stallings, 1998, 37) Uusi SNMPv2 paransikin mahdollisuutta kohdistaa kyselyjä useisiin objekteihin tai muuttujiin ja lisäsi joitakin uusia tietotyyppejä tiedon esittämiseen, vaikka parannusta tietoturvallisuuteen ei vielä tähän versioon saatukaan (Stallings, 2003, 780).

Esimerkiksi Ersue ja Claise (2011) toteavat, että SNMP:tä käytetään yleisesti virhetilanteiden ja suoritustehon seurantaan, sekä määriteltyjen toimintojen toiminnallisuuden seurantaan, epäjatkuvuuskohtien havainnointiin (esimerkiksi käytettävyysaika nollaantuu). Laskurit myös mahdollistavat tilastotiedon keräämisen eri valmistajien laitteista. (Ersue & Claise, 2011, 8–9)

SNMP:n avulla siis voitaisiin rakentaa luvussa 2 ideoitu verkon ”sääkartta”, jossa eri valmistajien laitteista kerätty tieto esitellään yhdellä ruudulla. SNMP:n tarjoama havaintotieto ei kuitenkaan ole siinä mielessä kuvailevaa, että se ei kerro esimerkiksi verkkoliikenteen sisällöstä tai liikennöintiin käytettävistä protokollista mitään. Sen avulla ei myöskään voida selvittää esimerkiksi sitä, mihin kohdeosoitteeseen liikenne on matkalla tai mistä se tulee. Siksi myöskään kuvan 2 kuormituspiikin syistä ei voida sanoa kovin tarkasti mitään. Tämän vuoksi selvittelyä havaintotiedon keruumenetelmistä täytyy jatkaa.

3.2. SYSLOG

BSD SYSLOG (RFC 3164) on alun perin Unix-käyttöjärjestelmissä käytetty protokolla välittää tila- tai lokiviestejä (log). SNMP:n tapaan BSD SYSLOG on yksinkertainen ja helppo käyttää, sillä perinteisesti BSD SYSLOG -viestillä on tarkoitettu tarkemmin määrittelemätöntä tekstipohjaista merkkijonoa ja kukin so-

vellus on voinut määritellä itse, millainen sisältö viestissä on. Koska viesti on välitetty tekstinä, ylläpitäjän on helppo tulkita sen sisältö, mutta standardin puute on hankaloittanut merkittävästi viestien automaattista käsittelyä. Vuonna 2009 IETF standardoi uuden IETF SYSLOG -protokollan (RFC 5424, 2009) lisäämällä muun muassa kentät aikaleimalle, lähettävän isäntäkoneen nimelle, lähettävän sovelluksen nimelle ja sanoman tunnisteelle. Näiden lisäysten tarkoituksena on parantaa suodatusmahdollisuuksia, yhteentoimivuutta ja vastaavuutta samankaltaisten sovellusten kesken. Lisäksi IETF on määritellyt IETF SYSLOG -protokollaan myös uusia ominaisuuksia, kuten tavan määritellä valmistajakohtaisia tietoelementtejä, mahdollisuuden käyttää yhteydellistä siirtotietä, salausta, tai tilaviestien kuittauksen. Salauksella tarkoitetaan sitä, että lähetettävä tilaviesti salakirjoitetaan niin, ettei sitä ole mahdollista tulkita ilman salausavainta. Yhteydellinen siirtotie ja tilaviestien kuittaus on käsitelty jäljempänä. (Ersue & Claise, 2011, 13—14; RFC 5424, 2009.)

BSD SYSLOG -protokollan merkitys Ersuen ja Claisen (2011) jaottelussa ei kenties tunnu ilmeiseltä, sillä myös SNMP tukee tilanmuutosviestien lähetystä. SNMP:n tilanmuutosviestit on kuitenkin tarkoitettu nimensä mukaisesti välittämään tietoa tapahtuneista muutoksista (joihin mahdollisesti toivotaan ylläpidon reagointia), joten ne eivät sinänsä sovellu esimerkiksi ilmoitusluontoisen tiedon välittämiseen. BSD SYSLOG -protokolla sen sijaan on suunniteltu kaikenlaisen tiedon välittämiseen, joten se täydentää havaintotiedon saatavuutta SNMP:n rinnalla. Tällöin esimerkiksi juuri syntyneestä häiriötilanteesta voitaisiin saada tieto verkkolaitteen lähettämän tilanmuutosviestin avulla ja myöhemmin selvitellä häiriön syytä BSD SYSLOG -protokollalla välitetyistä tilaviesteistä tarkastelemalla, millaisia viestejä laite on lähettänyt ennen häiriötä. Eräs käyttökelpoinen käyttöskenaario on myös tilanteet, joissa langattoman verkon tukiasema on vioittunut. Langattoman verkon luonteeseen kuuluu, että siihen langattomasti yhteydessä olevat päätelaitteet vaihtavat asiakkuutensa toiseen tukiasemaan esimerkiksi käyttäjän vaihtaessa paikkaa. Tällöin SYSLOG:n avulla langattoman verkon tukiasemat voivat lähettää tiedotusluonteisia tilaviestejä, joiden sisältönä on jonkinlainen päätelaitteen identifioiva tekijä ja siihen liittyvä toimenpide (esimerkiksi liittyminen tai poistuminen). Näin käyttäjien vihjeen perusteella päästään ongelmien jäljille.

Sekä BSD SYSLOG että SNMP on suunniteltu yksinkertaisiksi ja niihin tarpeisiin, joita suunnitteluhetkellä oli. Tämän vuoksi niissä on nykymittapuulla tarkasteltuna muutamia puutteita, joista merkittävimmät koskevat tietoturvallisuutta ja yhteydettömyyttä. Tietoturvallisuuden osalta SNMPv1 tai SNMPv2 eivät tue yhteyden salausta tai tietojen kyselijän käyttäjäpohjaista tunnistautumista. Tunnistautuminen SNMPv1 ja SNMPv2:ssa perustuu siihen, että kyseli-

jän tulee tietää oikea yhteisön nimi (community), joka yksinkertaistettuna vastaa salasanaa ja kyselyn tulee saapua IP-osoitteesta, jolla on määritelty oikeus kysyä tietoja. Koska yhteys ei ole salattu, kaikki välitettävä tieto, ”salasana” mukaan lukien, voidaan poimia verkkoliikenteen seasta selväkielisenä. Luonnollisesti ongelmaa voidaan kiertää sillä, että esimerkiksi verkkolaitteet tyypillisesti sijaitsevat omassa niin sanotussa hallintaverkossaan, johon pääsy on rajoitettu ja tällöin myös riski tietojen päätymisestä vääriin käsiin pienenee.

Protokollan yhteydellisyydellä tarkoitetaan sitä, että kahden osapuolen välinen yhteys avataan ja suljetaan ennalta sovitulla tavalla ja tällöin molemmat osapuolet tietävät yhteyden tilan. Osapuolten toisilleen lähettämät paketit numeroidaan ja niille lasketaan tarkistussumma, jolloin vastaanottaja tietää niiden lähetysjärjestyksen ja kuittaa jokaisen saapuneen paketin. Paketteja voi myös eri syistä hukkua matkalle, ja tällöin tietyn ajan kuluessa vastaanottaja voi pyytää lähettämään uudestaan tietyn paketin, jolloin vastaanottaja voi varmistua siitä, että kaikki lähetetyt paketit ovat saapuneet perille. Tarkistussumman avulla vastaanottaja voi varmistua siitä, ettei paketin sisältö ole muuttunut matkalla. Eräs yleisimmistä IP-verkon protokollista on Transmission Control Protocol (TCP), joka on yhteydellinen sisältäen nämä mainitut toiminnallisuudet.

Yhteydettömässä protokollassa vastaanottoa ei mitenkään valmistella, eivätkä paketit sisällä järjestysnumeroa. Tällöin vastaanottaja ei voi tehdä päätelmiä pakettien järjestyksestä eikä siitä, ovatko kaikki lähettäjän tarkoittamat paketit saapuneet perille. Tarkistussumman avulla vastaanottaja voi tunnistaa jollakin tavalla vioittuneen paketin, mutta sillä ei kuitenkaan ole keinoa pyytää lähettäjää lähettämään pakettia uudelleen. IP-verkossa yleisin yhteydetön protokolla on User Datagram Protocol (UDP). Sekä TCP:tä että UDP:tä kutsutaan kuljetuskerroksen protokolliksi, sillä ne nimensä mukaisesti kuljettavat esimerkiksi sovelluksen lähettämän tiedon määränpäähänsä.

Protokollaa, jonka tehtävänä on välittää tietynlaista tietoa, kuten SNMP välittää tietoa esimerkiksi laskureista tai BSD SYSLOG tilaviestejä, kutsutaan sovelluskerroksen protokollaksi. Sovelluskerroksen protokolla puolestaan hyödyntää jotakin kuljetuskerroksen protokollaa viestin välitykseen. Erilaisia kerroksia on toki muitakin, mutta niiden läpikäynti ei ole tämän tutkielman kannalta olennaista.

SNMP-protokollan versiot 1 ja 2 sekä BSD SYSLOG -protokolla käyttävät UDP:ta kuljetuskerroksessa. Tämän vuoksi niiden eräs haaste on yhteydettömyys: BSD SYSLOG:n tai SNMP:n tilanmuutosviestien saapumista vastaanottajalle ei kuitata, joten lähettäjällä ei ole mitään keinoa selvittää, saavuttiko lähetetty viesti vastaanottajan. Tällöin on mahdollista, että esimerkiksi häiriötilan-

teen vuoksi lähetetty SNMP-tilanmuutosviesti katoaa matkalle, eikä tieto häiriöstä koskaan saavuta ylläpitoa.

IETF SYSLOG -protokolla (RFC 5424, 2009) ja SNMP-protokollan versio 3 (RFC 3411, 2002) tuovat parannusta muun muassa yhteydettömyyden ongelmaan ja puutteelliseen tietoturvallisuuteen niin, että ne mahdollistavat yhteydellisen tiedonvälityksen, tiedon salauksen ja SNMPv3:n tapauksessa myös käyttäjäkohtaisen tunnistautumisen. On kuitenkin huomattavaa, että yhteydellisyys voidaan toteuttaa myös sovellustasolla: vaikka itse protokolla ei ole yhteydellinen, sovellus voidaan rakentaa niin, että sille lähetetyn viestin sisällössä tuodaan tavalla tai toisella ilmi se, että yhtään pakettia ei ole hukkunut ja yhteys sovellusten välillä on olemassa. Yhteyden ylläpito on kuitenkin sovelluksen vastuulla.

Vaikka IETF SYSLOG esittelee joukon kaivattuja uudistuksia, se hyväksyttiin standardiksi vasta vuonna 2009, joka tekee siitä historiallisessa perspektiivissä sangen uuden protokollan. Tämän vuoksi kestänee vielä tovi, ennen kuin erilaiset laitteet ja sovellukset toteuttavat sen vaatimukset täysimääräisesti. Kestänee toinen tovi, ennen kuin sovellus- ja laitekanta uusiutuu niin, että kaikki tilaviestit voidaan siirtää IETF SYSLOG:lla. Tätä ennen esimerkiksi verkonvalvonnassa on varauduttava ottamaan tilaviestejä vastaan sekä BSD SYSLOG:lla että IETF SYSLOG:lla. Tämä puolestaan luo haasteita tulkinnalle, kun muistetaan, että ensin mainittu ei aseta mitään sääntöjä sille, miten tilaviesti pitäisi rakentaa.

Tässä vaiheessa on siis käsitelty tapa hankkia kvantitatiivista tietoa SNMP:llä esimerkiksi verkkoliikenteestä ja toisaalta erilaisten sovellusten tai laitteiden välittämät tilaviestit (BSD tai IETF SYSLOG). Näiden perusteella ei kuitenkaan voida vielä sanoa mitään siitä, millaista liikenne on: mistä se tulee, mihin se menee, kuinka paljon informaatiota siirtyy tai kuinka kauan yhteys on ollut muodostuneena. Tähän pulmaan yritetään seuraavaksi etsiä ratkaisua.

3.3. IP flow information export ja packet sampling

Tietoverkon IP-liikenne koostuu yhteyksistä (flow), jotka kulkevat verkon läpi kahden verkkoon liitetyn laitteen välillä tiettynä aikana. Tiettyyn yhteyteen, tai vuohon, kuuluvilla paketeilla on tästä syystä yhteisiä ominaisuuksia, jotka voidaan tunnistaa havaintopisteessä (observation point). Kyseessä ei siis ole yhteydellisyydestä tai yhteydettömyydestä, kuten esimerkiksi TCP:n tai UDP:n tapauksessa, vaan esimerkiksi kahden laitteen välillä kulkevat yhteydettömän(kin) protokollan viestit (esimerkiksi SNMP tai BSD SYSLOG) muodostavat havaintopisteessä havaittavan IP-pakettien vuon kahden laitteen välillä ja tämän vuon ominaisuuksia voidaan seurata. (RFC 3917, 2004.)

IP flow information export (IPFIX) on muun muassa RFC 3917:ssa (2004) määritelty protokolla, joka luo puitteet siihen, miten havaintotietoa yhteyksistä havaitaan, mitataan, viedään (export) ja miten vietyä tietoa otetaan vastaan. Havaintotietoa yhteyksistä kerätään havaintopisteessä, joita voi olla yksi tai useampia ja havaintopisteenä voi toimia esimerkiksi reititin, jonka kautta liikenne kulkee. Havaintopisteessä muodostuu yhteystietueita (flow record), jotka sisältävät yhteyden ominaisuuksia kuten lähde- ja kohdeosoitteen sekä yhteydestä laskettuja suureita, kuten kaikkien yhteyden aikana liikennöineiden pakettien yhteenlasketun tavumäärään. Havaintopiste vastaa siitä, että yhteystietue luodaan, päivitetään, välitetään eteenpäin ja poistetaan tietyn ajan jälkeen, kun yhteys ei ole ollut aktiivinen. Yhteystietueet välitetään keräilijälle (collector), joka on tyypillisesti jokin ulkoinen järjestelmä, jossa esimerkiksi tietueet tallennetaan myöhempää käyttöä varten.

IPFIX:in avulla siis saadaan kerättyä tarkempaa tietoa siitä, millaista liikennettä tietoverkossa on. Kerätyistä tietueista voidaan luoda yhteenvetoja, joista selviää kunkin yhteyden kesto ja kellonaika, siirretty tavumäärä ja purskeisuus eli miten siirretty tavumäärä jakautuu koko yhteysajalle. Tietueista voidaan selvittää myös esimerkiksi yhteyksissä käytettyjä protokollia ja verkon keskeisiä palveluja. Tietueista haettavaa tietoa ei ole sinänsä ennalta määritelty, eli keräilijälle tallennetusta tiedosta voi tehdä millaisia hakuja hyvänsä. Eräs sovellusmahdollisuus onkin käyttää tietueiden sisältämää tietoa tunkeutumisen havainnointiin (intrusion detection), jossa voidaan etsiä esimerkiksi ennalta määrättyjä kohteita tai tietynlaista yhteyskäyttäytymistä. Tunkeutumisen havainnointiin toki liittyy joukko muitakin havainnointimenetelmiä, jotka eivät kuulu tämän tutkielman piiriin.

Havaintopisteet lienee järkevää sijoittaa sellaisiin tietoliikenneverkon kohtiin, joiden kautta kulkee mahdollisimman iso osa verkon liikenteestä. Tällaisia kohtia voisivat olla esimerkiksi internet-liityntäpiste ja keskeiset reitittimet. Liikenteen analysointi kuluttaa resursseja, joka voi esimerkiksi reitittimen tapauksessa heikentää kykyä suorittaa pääasiallista tehtävää eli liikenteen välittämistä. Valmistaja voi huomioida analysoinnin tehontarpeen liittämällä, tai myymällä, laitteistoonsa esimerkiksi erillisen prosessointimoduulin, joka on suunniteltu erityisesti liikenteen analysointiin. IPFIX kuitenkin mahdollistaa sen, ettei jokaista havaintopisteen läpi kulkevaa pakettia välitetä yhteyksien mittausprosessille, vaan paketeista voidaan valita tietty osajoukko (esimerkiksi joka sadas, jokainen TCP-paketti, satunnaisuuteen perustuva joukko jne.) ja käyttää vain tätä näytteistettyä osajoukkoa yhteystietueina. Myös tieto käytetystä näytteistyksestä välitetään keräilijälle yhteystietueessa. (RFC 3917, 2004.)

IPFIX ei aseta havaintopisteen mittausprosessille mitään vaatimuksia siitä, millaisia näytteistysmenetelmiä siinä tulisi olla toteutettuna. Tämän vuoksi vuonna 2002 alettiin kehitellä protokollaa, jonka tavoitteena oli muun muassa määritellä käytettävät näytteistysmenetelmät ja ne tiedot, jotka näytteistetyistä paketeista raportoidaan eteenpäin keräilijälle. Vuonna 2009 esiteltiin Packet Sampling -protokolla (PSAMP), joka määrittelee joukon näytteistysmenetelmiä. Näitä ovat esimerkiksi systemaattinen pakettien määrään perustuva näytteistys (esimerkiksi joka sadas), systemaattinen aikaan perustuva näytteistys (kuten pakettien määrään perustuva näytteistys, mutta näytteistetään paketteja aikavälein lukumäärän sijaan), erilaisia todennäköisyyksiin perustuvia näytteistyksiä, paketin ominaisuuksien perusteella tehtävä näytteistys tai paketin sisällöstä laskettaviin tiivisteisiin (hash) perustuva näytteistys. (Claise & Wolter, 2007, 212) Näytteistysmenetelmien tarkempi esittely on kuitenkin rajattu tämän tutkielman ulkopuolelle.

IPFIX on siis yleinen yhteystietueiden vientiin suunniteltu protokolla, jonka vahvuus on välittää yhteyksiin liittyvää (usein kumulatiivista) tietoa. PSAMP taas määrittelee joukon menetelmiä, joilla voidaan näytteistää paketteja ja välittää tietoa kunkin yksittäisen paketin ominaisuuksista. Tällöin IPFIX ja PSAMP tukevat toisiaan: verkossa voidaan IPFIX:in avulla seurata yleisemmin IP-liikennettä ja PSAMP:in avulla valita tietty osajoukko liikenteestä tarkempaan analyysiin. Vaikka IPFIX ja PSAMP soveltuvat erilaisiin tarpeisiin, Ersue ja Claise (2011, 15—18) käsittelevät niitä yhdessä. Syy on hätkähdyttävän yksinkertainen, kuten Claise ja Wolter (2007, 213) toteavat, sillä IPFIX ei tee eroa sen välillä, onko välitetyssä yhteystietuessa tietoa yhdestä vai useammasta paketista. IPFIX on nimittäin suunniteltu yleiskäyttöiseksi protokollaksi havaintotiedon vientiin, jolloin PSAMP:illa näytteistetty yksittäinen paketti voidaan ajatella IPFIX-protokollan näkökulmasta yksittäiseksi yhteystietueeksi. Tällöin PSAMP voi hyödyntää IPFIX:issa määriteltyä tapaa viedä yhteystietueet keräilijälle, jolloin myös sen toteutus muuttuu valmistajan näkökulmasta yksinkertaisemmaksi.

4. Havaintotiedon käytöstä

Tiedonkeruumielessä luvussa 3 esitellyt mekanismit antavat tuiki erilaista tietoa kukin: esimerkiksi kohdassa 3.1 esitelty SNMP mahdollistaa hyvin laajan, alueen, jolta tietoa voidaan kerätä. Vaan mitä tietoa pitäisi kerätä ja millaisella frekvenssillä? Esimerkiksi verkkolaitteiden kuormitus saattaa vaihdella hetkellisesti, jolloin esimerkiksi viiden minuutin keskiarvo saattaa kadottaa kuormituspiikkejä. Toisaalta liian innokas ja laaja-alainen tiedonkeruu saattaa prosessointiteholtaan vaatimattomammissa laitteissa viedä suhteettomasti prosessoin-

tiaikaa. Keruu asettaa myös vaatimuksia tiedonkeruuta hoitavalle laitteistolle esimerkiksi tehontarpeen ja käytössä olevan levytilan suhteen. Esimerkiksi viiden minuutin välein tehtävät keruuprosessit tulee saada valmiiksi ennen seuraavan keruukierroksen alkua ja jokainen tallennettu havaintoarvo kuluttaa käytössä olevaa levytilaa. Siksi tehokaskin laitteisto voi osoittautua riittämättömäksi, jos mitoitusta ei huomioida järjestelmää suunniteltaessa.

Kohdassa 3.2 esitelty SYSLOG tukee SNMP:n avulla kerättyä tietoa tilaviestein. Tilaviestit kuitenkin kerätään tyypillisesti eri paikkaan kuin SNMP:llä kerätty havaintotieto, sillä tilaviestejä voidaan haluta vastaanottaa verkkolaitteiden lisäksi myös esimerkiksi Unix-pohjaista käyttöjärjestelmää käyttäviltä palvelimilta (ja työasemilta). Vaikka tilaviestit vastaanotettaisiinkin samaan paikkaan SNMP:llä kerätyn havaintotiedon kanssa, ongelmaksi muodostuu se, miten yhdistää helposti SNMP-havaintotieto ja tilaviestit. SNMP:llä kerätty tieto täytynee sijoittaa jonkinlaiselle aikajanalle ja etsiä tältä aikajanalta poimittuja aikaleimoja vastaavat merkinnät tilaviesteistä. Tällöin aikaleimojen tulee vastata mahdollisimman hyvin toisiaan, toisin sanoen laitteiden kellot tulisi pitää mahdollisimman hyvin ajassa (tähän on toki menetelmiä, esimerkiksi RFC 958:ssa määritelty Network Time Protocol, mutta niiden käsittely on tämän tutkielman aiheen ulkopuolella). Tilaviestien tarkastelu pienellä aineistolla on suhteellisen helppoa. Mutta jos SNMP:n havaintotiedon tarkasteluväli on oletuksellinen viisi minuuttia, aikahaarukka tarkasteltaville tilaviesteille on myös viisi minuuttia. Tarkasteltavien tilaviestien määrä voi olla suhteellisen iso, jos tilaviestejä saapuu esimerkiksi keskeiseltä reitittimeltä. Tällöin tilanmuutosviestit voivat auttaa paikallistamaan ongelman syntyhetkeä paremmin.

Kun SNMP ja SYSLOG tarjoavat tietoa esimerkiksi yhteyksistä eri verkkolaitteiden liityntöjen (interface) välillä, kohdassa 3.3 esitellyt IPFIX ja PSAMP syventävät SNMP:lla ja SYSLOG:lla kerättyä havaintotietoa tarjoamalla yksityiskohtaisempaa tietoa esimerkiksi tietyllä hetkellä havaintopisteessä kulkevasta liikenteestä. IPFIX:in ja PSAMP:in avulla kerätty havaintotieto on myös luonteeltaan yksilöivää, sillä se paljastaa verkkoon liitettyjen laitteiden keskinäiset yhteydet. Sen vuoksi näiden protokollien avulla kerätyn havaintotiedon säilytykseen tulee kiinnittää erityistä huomiota. Havaintotietoa myös kertyy, riippuen näytteistystiheydestä, ripeästi, joten tallennusresurssien oikea mitoitus on tärkeää. Pakettien ja yhteyksien tarkempi tarkastelu siis lisää SNMP:n ja SYSLOG:n oheen vielä kolmannen kokonaisuuden. Tämä osaltaan lisää käytettävissä olevan havaintotiedon määrää suorassa suhteessa havaintopisteiden ja verkossa kulkevan liikenteen määrään.

Tuntuisi myös järkevältä yrittää valjastaa automatiikka hälyttämään mahdollisista häiriöistä, sillä esimerkiksi yksittäisten yhteysvälien tai liityntöjen

ongelmat eivät välttämättä näy yleiskuvissa (ks. esim. kuva 1). Tämä kuitenkin vaatii riittävän kattavan seurannan ohella käsityksen siitä, mikä on normaalia liikennettä ja tietenkin suunnitelmaa siitä, mitä halutaan seurata. Todennäköisesti raja-arvojen asettaminen vaatii jo itsessään seurantatietoa ja jatkuvaa hienosäätöä, jotta ylläpito ei vastaanota toistuvia vääriä hälytyksiä.

”Verkkosääkartan” kaltaisissa yleiskuvissa on huolehdittava siitä, että kuva kattaa riittävän suuren osan sen esittämästä verkosta ja sisältää riittävän määrän mittauspisteitä. Häiriötilanteessa verkkoliikenteen kulkua pitäisi voida seurata mahdollisimman luotettavasti: liikennettä ei saa ”kadota” verkosta tai tulla lisää jostakin, esimerkiksi kiertämällä sellaisen laitteen kautta, josta ei kerätä havaintotietoa.

Jatkoa ajatellen kerätty havaintotiedon sisältämät elementit täytynee myös dokumentoida ja luokitella, jotta kerättyä informaatiota voidaan hyödyntää myöhemmin ilmenevissä tarpeissa. Näin ehkä voitaisiin myös tunnistaa sellaisia järjestelmän osia, joista ei vielä kerätä havaintotietoa. Myös käyttäjiä kiinnostanee yhä enemmän, millaista tietoa heistä tallennetaan, joten kokonaisuuden kannalta on viisasta osata nimetä ne tiedon lajit ja varannot, joihin tietoa säilötään.

5. Erilaisia seurantajärjestelmiä

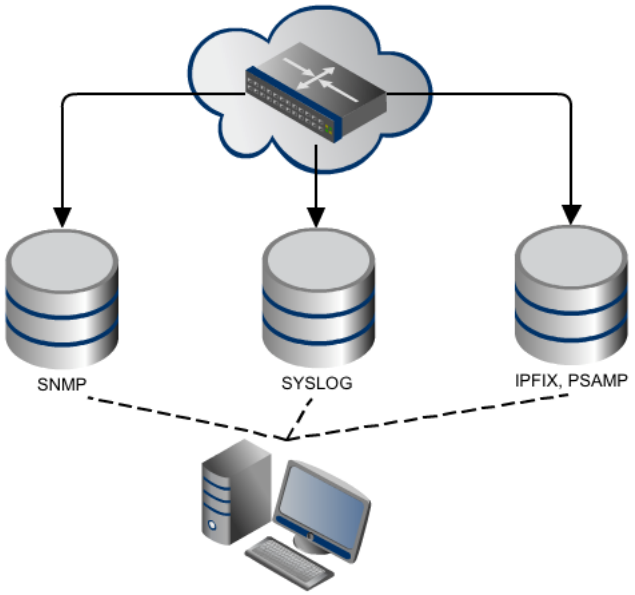
Seurantajärjestelmää suunniteltaessa eräs vaikuttava tekijä on se, miten isoa järjestelmää niillä on tarkoitus seurata ja miten yhtenäinen järjestelmä on esimerkiksi laitekantansa osalta. Tästä syystä isoihin kokonaisuuksiin, kuten esimerkiksi suuryrityksien tarpeisiin, laaditut järjestelmät sisältävät väistämättä erilaisia reunaehtoja siitä, millainen seurattavan järjestelmän tulee olla. Voidaan esimerkiksi ajatella, että ominaisuuksia ei voida täysipainoisesti hyödyntää, jos seurattavat laitteet eivät toteuta kaikkia seurantajärjestelmän toimintoja. Toisaalta seurantajärjestelmät saattavat painottua tiettyjen laitevalmistajien tuotteisiin ja laitevalmistajat ovat kehitelleet omia seurantatuotteitaan, jotka luonnollisesti toimivat parhaiten juuri heidän laitteittensa kanssa.

Voi myös olla, että valmistajat ovat kehitelleet laitteistoihinsa omia menetelmiä havaintotiedon keruulle. Toisinaan tällainen aktiivisuus on hyvästä, sillä esimerkiksi IPFIX:n perustalle on otettu Cisco Systemsin kehittämä ja vapaaseen käyttöön julkaisema Cisco NetFlow -protokolla (Claise & Wolter, 2007, 201—202). Kaikki valmistajakohtainen kehitystyö ei kuitenkaan saa standardin asemaa, vaan jää taka-alalle. Tällöin tiettyjä keruumenetelmiä hyödyntävä seurantajärjestelmä saattaa olla sidoksissa ja toimia kunnolla vain tietyn valmistajien laitteiden kanssa. Tämä on toki valmistajille edullinen tilanne, mutta pitemmän päälle laitehankinnoissa valinnan mahdollisuutta kaipaavalle kiusallinen

puhumattakaan tilanteista, joissa laitevalmistaja sulautuu toiseen, tai sen toiminta syystä tai toisesta lakkaa.

Erilaisten lähteiden tuottaman tiedon keskittäminen luo kysymyksen siitä, kenellä on pääsy tietoon (ks. esim. Valtiovarainministeriö, 2010, 21). Tämä tarve hallita tietoa puolestaan saanee aikaan sen, että isommissa seurantajärjestelmissä havaintotiedon luottamuksellisuuden ja eheyden hallinta on hyvin hienojakoista. Niinpä jos järjestelmää käyttää muutama henkilö, voi tuntua erikoiselta asentaa esimerkiksi kokonaan oma palvelimensa identiteetin hallintaa varten. Toisaalta pienehkön ympäristön seurantaan suunniteltu järjestelmä voi osoittautua tältä osin puutteelliseksi.

Työskentely laajan kokonaisuuden kanssa pakottaa ohjelmiston käyttäjän tiettyyn kurinalaisuuteen ja loogiseen jaotteluun ohjelman sisällä. Tämä voi näkyä esimerkiksi seurantaohjelmiston lisäkomponenttien kirjona, syvinä valikkorakenteina tai tarpeettoman raskaana hierarkiana. Käytön kankeus voisi siksi johtaa turhautumiseen ja haluttomuuteen hyödyntää ohjelmistoa täysipainoisesti.



Kuva 3. Kokonaisvaltainen seuranta pyrkii hyödyntämään kaikkia havaintotiedon lähteitä.

Yleensä kukin seurantatiedon keruujärjestelmä sisältää havaintotietoa vain yksittäisestä havaintotiedon lajista. Tällöin tiedon yhdistely on haastavampaa, sillä kokonaisuuden eri osat ovat hajautuneet useaan eri paikkaan (ks. kuva 3). Niinpä tiedon tarvetta palvelee mahdollisimman monesta järjestelmästä saatavan tiedon keskittäminen yhteen paikkaan, josta voidaan hakea kulloinkin haluttua tietoa.

Keskittämisellä tähdätään myös siihen, että eri lähteiden havaintotietoa voidaan yhdistellä, jolloin niiden välisiä syy-seuraussuhteita voidaan helpommin selvitellä. Ikävä kyllä Shiravi, Shiravi ja Ghorbani (2011) toteavat, että monet havaintotiedon visualisointiin tarkoitetut järjestelmät on suunniteltu visualisoimaan yksittäistä havaintotiedon tyyppiä, eli esimerkiksi vain SNMP:llä saatua tietoa. Myös Liao, Blaich, VanBruggen ja Striegel (2010) toteavat, ettei tietoliikenneverkon päivittäisseurantaan luodut yksittäiset työkalut juurikaan edistä kokonaisuuden hahmottamista.

Liao ja muut (2010) ovat luoneet sovelluksen nimeltä ENAVis (Enterprise Network Activities Visualization), jonka tavoitteena on havainnollistaa yhteyksiä palvelimen, käyttäjän ja sovelluksen välillä niin, että palvelinten välisen tietoverkon valvontatietoa hyödyntämällä saadaan mukaan myös palvelinprosessien välinen viestintä. Sovellus hyödyntää IPFIX:in pohjalla olevaa NetFlow -protokollaa, SNMP:tä ja palvelimilla sekä työasemilla ajettavia valvontaprosesseja. (Liao et al., 2010.) Kirjoittajien esittämä järjestelmä kuitenkin havainnollistaa myös sen, ettei esimerkiksi verkon ylläpito voi yksin vastata kaikkiin tietoturvallisuuden kokonaisvaltaiseen ylläpitoon liittyviin kysymyksiin, vaan ylläpito vaatii yhteistyötä eri osa-alueiden ylläpitäjien kanssa. Kirjoittajien järjestelmän heikkous on myös se, että järjestelmä on suunniteltu Unix-pohjaisiin järjestelmiin, jolloin esimerkiksi Windows- tai muut käyttöjärjestelmät jäävät havaintotiedon keruun ulkopuolelle.

Shiravi ja muut (2011) puolestaan esittelevät kattavan joukon erilaisia yksittäisiä seurantajärjestelmiä. Seurantajärjestelmät on jaoteltu teemoittain ja antaa kuvan erilaisista tavoista esittää havaintotietoa. Siksi esittely toimii nähdäkseni hyvänä pohjana seurannan suunnittelulle.

6. Tulkintakyvystä ja oppimisesta

Useat kasvatustieteen tutkijat ovat esittävät, että koulutus ja oppiminen jakautuu kolmeen lajiin: muodolliseen (formal), epämuodolliseen (non-formal) ja informaaliin (informal). Muodollinen koulutus vastaa asteittain etenevää kasvatusjärjestelmää, joka kiinnittyy tunnustettuihin tutkintohierarkioihin. Tällaisella tarkoitetaan esimerkiksi peruskoulun luokkia tai yliopistotutkintoa. Epämuodollinen koulutus puolestaan perustuu usein organisaation tai yksilön omiin koulutus- tai oppimistarpeisiin. Epämuodollisen koulutuksen pääpaino yleensä keskittyy taitoihin, kuten jonkin järjestelmän käytön opettelu. Tämän voinee ymmärtää niin, että epämuodollinen koulutus tähtää mahdollisimman välittömään hyötyyn esimerkiksi yksilön tai organisaation näkökulmasta toisin kuin muodollisessa koulutuksessa. Informaalilla oppimisella tarkoitetaan ”piilevää” oppimista, jolla ei ole muodollisen tai epämuodollisen oppimisen kaltaista tie-

dostettua oppimisen tavoitetta. Sen vuoksi informaali oppiminen tapahtuu yleensä muun toiminnan sivutuotteena. Tällaista oppimista voisi olla esimerkiksi oppiminen erehdyksestä, olettamuksista, kokemuksista ja niin edelleen. Mainitut muodot lomittuvat monin tavoin keskenään, joten niiden erottaminen toisistaan on käytännössä vaikeaa. (Rinne & Salmi, 2005, 149—151)

Tulkinta on nähdäkseni sivistynyt arvaus, sillä totuus rakentuu tulkitsijan käytössä olevien tietojen perusteella. Niinpä onnistuneessa tulkinnassa tarvitaan kaikkia oppimisen muotoja: muodollinen oppiminen luo teoreettisen tietoperustan, jonka avulla tulkitsija voi ymmärtää havaintojen syntymekanismeja. Epämuodollinen oppiminen puolestaan laajentaa tulkitsijan teoreettista tietoperustaa esimerkiksi silloin, kun tulkitsija syventää omaa osaamistaan opiskelemalla omaehtoisesti uusia teknologioita. Piilevä oppiminen nähdäkseni integroi epämuodollista ja muodollista oppimista niin, että esimerkiksi virheen seurauksena syntyy käytännön esimerkkejä tilanteista, jotka ovat ymmärrettävissä muodollisen ja epämuodollisen oppimisen kautta saaduilla tiedoilla. Tällaisia informaaleja oppimistilanteita voisi syntyä esimerkiksi häiriötilanteissa, joiden selvittely nivoo yhteen teoriatiedon (miksi häiriö syntyy) ja taidollisen tiedon (miksi häiriö ilmenee tavalla, jolla se ilmeni).

Lienee myös selvää, että kukin oppii omassa tahdissaan ja itselleen luontevimmilla tavoilla. Yhtä lailla lienee informaalin oppimisen käsitteen kautta selvää, että eri ihmiset hahmottavat ja nivovat yhteen asioita erilaisin tavoin. Niinpä eräs toive havaintotiedon visualisoinnissa voisi olla myös mahdollisuus muodostaa omaa oppimista ja havainnointia tukevia kokonaisuuksia, kuten sovelluksen henkilökohtaiset työpöydät tai muutoin muokattavat näkymät. Vapaasti muokattavat kokonaisuudet myös antavat tilaa innovatiiviselle tietojen yhdistelylle, havainnoinnille ja palvelevat erilaisten määriteltyjen asiakokonaisuuksien seurantaa.

Erilaiset oppimisen mallit ja ammattitaidon kehittyminen työtä tekemällä ansaitsisivat kokonaan omat lukunsa. Aihepiirin laajuuden vuoksi ne on rajattu esimerkiksi hahmottamiseen liittyvien kysymysten ohella tämän tutkielman käsittelyn ulkopuolelle. Mainittujen tekijöiden vaikutus on kuitenkin tärkeää huomioida valvontajärjestelmiä suunniteltaessa. Siten oppimista muodosta riippumatta on pyrittävä tukemaan kaikin käytettävissä olevilla keinoin.

7. Tulevaisuuden haasteista

Kuten luvussa 2 tuli ilmi, tietoverkon ylläpitoon liittyy yhä enemmän myös tietoturvallisuus eri muodoissaan. Tietoturvallisuuden haasteena on pysytellä askel teknisen kehityksen edellä tai vähintään seurata tiiviisti kehityksen kannoilla. Uudet protokollat ja sovellukset hyödyntävät verkkoa tavoilla, joita

suunnittelussa ei välttämättä ole huomioitu. Tällöin myös esiin nousee ongelmia, jollaisia aiemmin ei ole osattu huomioida. Kuitenkin ongelmat tulisi ratkaista käytössä olevin keinoin.

Eräs haaste seurannalle on IP-verkon osoiteavaruutta laajentava protokolla IPv6 (RFC 2460, 1998), joka alkaa hitaasti yleistyä. Koska kyseessä on uusi protokolla, myös seurantatyökalut täytyy päivittää tukemaan sitä. Barrera ja van Oorschot (2009) toteavat, että myös rikolliset ovat päivittäneet työkalunsa tukemaan IPv6:a. Kuitenkin useat tietoverkon seurannan työkalut tukevat vain nykyisin käytössä olevaa IPv4:a (RFC 791, 1981). Barrera ja van Oorschot (2009) osoittavat, että esimerkiksi nykyiset visuaaliset keinot lähde- ja kohdeosoitteiden välisten yhteyksien esittämiseen ovat riittämättömiä. He kuitenkin antavat esimerkkejä vaihtoehtoisista visualisointimenetelmistä, jotka voisivat olla pohjana uusille ratkaisuille.

Ajan henkeen kuuluu käyttäjän liikkuvuus ja mobiilit päätelaitteet. Kuitenkaan langattomia verkkoja omine haasteineen ei tässä työssä käsitellä. Langattoman verkon luonteeseen kuuluu se, ettei asiakaslaitteella ole määriteltyä tukiasemaa. Tällöin ajatus esimerkiksi tukiasemien liityntöjen seuraamisesta SNMP:llä ei anna tietoa yksittäisen asiakkaan ongelmista. Toisaalta langattomille verkoille on olemassa valvontajärjestelmiä, jotka pitäisi saada vaihtamaan tietoa langallisten verkkojen valvontajärjestelmien kanssa.

Häiriötilanteissa syiden selvittely on hankaloituu, jos riittävää dokumentaatiota laitteiden sijainneista, verkon rakenteesta, verkkolaitteiden asetuksista sekä historiatietoa näihin tehdyistä muutoksista ei ole käytettävissä. Niinpä tietoverkon valvonta on myös dokumentaation ajantasaisuudesta huolehtimista. Kenties tulevaisuuden valvontajärjestelmä voisi sisältää osiot myös dokumentaatiolle ja muutosten hallinnalle?

8. Yhteenveto

Tässä tutkielmassa esiteltyjen protokollien avulla voidaan keräillä kohtuullisen laaja-alaiselta tuntuvaa tietoa esimerkiksi yhteysvälien kuormituksesta, eri laitteiden lähettämistä viesteistä, yhteyksistä ja pakettien sisällöistä. Tällä tiedolla voitaisiin rakentaa esimerkiksi luvussa 2 ideoitu verkon ”sääkartta”.

Kuitenkaan näiden menetelmien avulla ei voida sanoa mitään esimerkiksi suorituskyvystä tai latenssista, eli siitä, kuinka nopeasti isoja datamääriä voidaan liikutella verkossa tai kauanko paketin matka kestää lähteestä kohteeseen. Toistaiseksi ei osata sanoa myöskään mitään varmaa esimerkiksi joukkolähetyksestä (multicast) tai palvelujen saatavuudesta, eli siitä, kuinka kuormitettuja palvelimet ovat tai miten nopeasti ne vastaavat kyselyihin.

Tutkielmassa on myös tullut ilmi, että tietoa voidaan kerätä monella tavalla ja säilöä useassa paikassa. Tiedon hajanaisuus puolestaan luo haasteita sen hyödyntämiseen, sillä tieto on eri muodoissa ja sen hyödyntämiseen luodut järjestelmät ovat vielä pitkälti painottuneita yhden tietomuodon käsittelyyn. Niinpä ylläpidon työmäärää lisää se, että ongelmien selvittelyyn tarvitaan usean eri tavoin toimivan järjestelmän käyttö.

Kasvava tarve automatisointiin luo paineen pyrkiä automatisoimaan myös valvontaa. Tämä puolestaan luo haasteen havaintotietoa käsitteleville järjestelmille. Järjestelmien tulisi kyetä erottelemaan normaali liikenne poikkeavasta liikenteestä ja mukautumaan erilaisiin tilanteisiin niin, ettei ylläpito huku vääriin hälytyksiin. On myös huomattavaa, ettei automaattinen järjestelmä korvaa kokenutta tulkitsijaa, sillä viime kädessä tulkitsijasta riippuu, miten järjestelmän tuottamaa tietoa käytetään. Tulkitsijan kykyyn tulkita vaikuttaa nähdäkseni ainakin toimintaympäristön tuntemus (eli miten valvottava kokonaisuus on suunniteltu ja toteutettu), kerätyn havaintotiedon kokonaisvaltaisuus (eli kerätäänkö havaintotietoa yhdenmukaisesti ja riittävän laaja-alaisesti), käsitys hallintajärjestelmän käyttökelpoisuudesta, kokemus ja oppimiskyky. Inhimillisyyttäkään ei tule unohtaa: virheistä oppii.

Keskeistä lienee siis se, että käsitellyillä protokollilla on tarkoituksensa ja oma hyötynsä valvonnalle. Toimiva valvonta hyödyntää kaikkia käytössä olevia mekanismeja, pyrkii uudistumaan ja löytämään vaihtoehtoisia tapoja tehdä sama asia helpommin.

Viiteluettelo

Barrera, D., & van Oorschot, P. C. (2009). Security visualization tools and IPv6 addresses. *6th International Workshop on Visualization for Cyber Security 2009,* Atlantic City, NJ, USA. 21-26.

Claise, B., & Wolter, R. (2007). *Network management: Accounting and performance strategies*. Indianapolis, IN: Cisco Press.

CSC — Tieteen tietotekniikan keskus Oy. (2011). *Funet weathermap.* Saatavilla: http://www.csc.fi/funet/status/tools/wm (viitattu 8.12.2011).

Ersue, M., & Claise, B. (2011). *An overview of the IETF network management standards.* IETF Network Working Group. M. Ursue, (Ed.). Available at: http://tools.ietf.org/html/draft-ietf-opsawg-management-stds-02 (accessed 3 December 2011).

Klein, D. V., & Sellens, J. (2010). *#20: Running the numbers: System, network, and environment monitoring*. Berkeley, CA: The USENIX Association. Available at: http://techbus.safaribooksonline.com/book/networking/network-monitoring/9781931971706

Liao, Q., Blaich, A., VanBruggen, D., & Striegel, A. (2010). Managing networks through context: Graph visualization and exploration. *Computer Networks,* 54(16), 2809-2824.

Pietilä, V., Malmberg, T., & Nordenstreng, K. (2004). Teoreettisia yhtymäkohtia ja vastakkainasetteluja - katsaus Suomesta. Teoksessa I. Ruoho, & K. Nordenstreng (toim.), *Tiedotusopin peruskurssin lukemisto*, Tampereen yliopiston Tiedotusopin laitoksen opetusmoniste D46/2004.

RFC 791. (1981). *Internet Protocol*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc791 (accessed 7 December 2011).

RFC 1067. (1988). *A simple network management protocol*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc1067 (accessed 7 December 2011).

RFC 1441. (1993). *Introduction to version 2 of the Internet-standard Network Management Framework*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc1441 (accessed 7 December 2011).

RFC 2460. (1998). *Internet Protocol, Version 6 (IPv6) Specification*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc2460 (accessed 7 December 2011).

RFC 3411. (2002). *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc3411 (accessed 7 December 2011).

RFC 3164. (2001). *The BSD syslog Protocol*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc3164 (accessed 7 December 2011).

RFC 3917. (2004). *Requirements for IP Flow Information Export (IPFIX)*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc3917 (accessed 7 December 2011).

RFC 5424. (2009). *The syslog protocol.* IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc5424 (accessed 3 December 2011)

RFC 5474. (2009). *A Framework for Packet Selection and Reporting*. IETF Network Working Group. Available at: http://tools.ietf.org/html/rfc5474 (accessed 5 December 2011).

Rinne, R., & Salmi, E. (2005). *Oppimisen uusi järjestys* (5. painos). Tampere: Vastapaino.

Shiravi, H., Shiravi, A., & Ghorbani, A. (2011). A survey of visualization systems for network security. *IEEE Transactions on Visualization and Computer Graphics* 99. To appear.

Subramanian, M., Gonsalves, T. A., & Rani, N. U. (2010). *Network management: Principles and practice*. Pearson Education India.

Sähköisen viestinnän tietosuojalaki 16.6.2004/516.

Valtiovarainministeriö. (2010). *Sisäverkko-ohje* (VAHTI 3/2010). Saatavilla: http://www.vm.fi/vm/fi/04_julkaisut_ja_asiakirjat/01_julkaisut/05_valtion hallinnon_tietoturvallisuus/20101203Sisaeve/Sisaeverkko-ohje.pdf (viitattu 13.12.2011).

Vertaisverkot

Teemu Ruotsalainen

Tiivistelmä.
Vertaisverkoilla on paljon potentiaalia, joka on tällä hetkellä vielä suurilta osin käyttämättä. Vertaisverkko on suurin kilpailija asiakas-palvelin-mallille. Tässä tutkielmassa perehdytään yleisesti vertaisverkkoihin, niiden rakenteisiin, sovelluksiin ja tulevaisuuden mahdollisuuksiin.

Avainsanat ja -sanonnat: vertaisverkko, arkkitehtuurit
CR-luokat: D.2.11

1. Johdanto

Vaikka moni ajattelee ensimmäisenä vertaisverkoista puhuttaessa piratismia tiedostonsiirrossa, voisi niitä hyödyntää huomattavasti enemmän myös laillisessa toiminnassa. Vertaisverkkoja voidaan käyttää hyväksi monessa eri tilanteessa ja lisää käyttötarkoituksia keksitään jatkuvasti. Esimerkkejä nykypäivän tekniikoista ovat tiedostojen siirron lisäksi muun muassa suoratoisto, pilvilaskenta ja yhteydenpito, joista kerron lisää myöhemmin.

Tässä tutkielmassa selvennän ensin, mitä vertaisverkot oikeastaan ovat yleisellä tasolla, ja kuinka ne eroavat asiakas-palvelin-mallista. Sen jälkeen esittelen vertaisverkkojen arkkitehtuureita ja erityyppisiä sovelluksia, jotka hyödyntävät vertaisverkkoja. Lopuksi pohdin myös, miten vertaisverkkoja voitaisiin hyödyntää tehokkaammin.

2. Yleistä vertaisverkoista

Vertaisverkko on tietoliikenteessä käytettävä toimintamalli, joka on suurin kilpailija perinteiselle asiakas-palvelin-mallille. Asiakas-palvelin-malli on yleisin internet-sovelluksissa käytetty arkkitehtuuri. Siinä asiakkailla ja palvelimilla on omat roolinsa. Palvelimilla säilytetään kaikki sisältö ja palvelut, joita asiakkaille tarjotaan. Asiakkaat voivat ottaa yhteyden palvelimelle ja käyttää palvelimen tarjontaa, mikäli se on tavoitettavissa ja kykenee palvelemaan kaikki asiakkaita. Asiakas-palvelin-mallin toimintaa voivat rajoittaa internet-yhteyden hitaus,

prosessorin tehottomuus, hitaat tallennus- ja kirjoitusnopeudet sekä levytila. Palvelinten kaatuminen johtaa myös suuriin vaikeuksiin käytettäessä kyseistä arkkitehtuuria. [Liu and Antonopoloulos, 2009.]

Vertaisverkkojen suurimpia hyötyjä asiakas-palvelin-malliin verrattuna ovat vakaus, tehokkuus, skaalautuvuus sekä matalat kustannukset. Vertaisverkossa jokainen käyttäjä, eli solmu, toimii palvelimena ja asiakkaana. [Liu and Antonopoloulos, 2009.] Solmut ovat asiakkaita, kun ne käyttävät muiden verkossa olevien solmujen resursseja, ja puolestaan palvelimia, kun ne jakavat omia resurssejaan muille [Yan, 2009]. Tämä poistaa erillisten palvelinten käyttöön liittyvät ongelmat, sillä näin palvelimia tarvitaan paljon enemmän, mikä tekee verkosta huomattavasti vakaamman [Liu and Antonopoloulos, 2009]. Sen takia vertaisverkkoa on todella vaikea horjuttaa normaaleilla palvelunestohyökkäyksillä eikä esimerkiksi sähkökatkos välttämättä kaada sitä, koska yhden tai muutaman solmun puuttuminen ei vaikuta suuriin verkkoihin juuri ollenkaan. Kustannustehokkuutta edistää se, että vertaisverkossa ei tarvita normaaliin asiakas-palvelin-malliin verrattuna palvelimia läheskään yhtä paljon, tai ei jopa ollenkaan.

Ensimmäinen ”versio” internetistä (ARPANET) vuonna 1969 oli samalla myös ensimmäinen vertaisverkko. Siinäkin tavoitteena oli yhdistää solmut toisiinsa enemmänkin yhdenvertaisina kuin asiakkaana ja palvelimena. Vasta internetin käyttäjämäärien räjähdettyä ja sen kaupallistuttua 1990-luvulla asiakas-palvelin-malli yleistyi ja koko internetin luonne muuttui. [Minar and Hedlund, 2001.] Vaikka vertaisverkkoja on siis ollut olemassa varsin kauan, on niiden hyödyntäminen potentiaaliin nähden nykyään todella vähäistä. Vertaisverkkoja ovat hyödyntäneet muun muassa tässä tutkielmassa myöhemmin esittelemäni BitTorrent-yhteystapaa käyttävät ohjelmat ja niiden kautta tapahtuva tiedostojen jako, johon piratismikin liittyy. Ehkä yksi osasyy vertaisverkkojen sivuuttamiseen tietoliikenneratkaisuissa voisikin olla se, että yleisölle näkyvin esimerkki niistä on niinkin kiistanalainen kuin piratismi.

3. Arkkitehtuurit

Vertaisverkot jaetaan yleensä kolmeen kategoriaan, jotka ovat vanhimmasta uusimpaan: keskitetyt (centralized), keskittämättömät (decentralized) ja hybridit (hybrid) verkot. Arkkitehtuureissa käytetyt etsintämenetelmät (search systems) ovat rakenteellinen (structured) ja rakenteeton (unstructured). [Vu et al., 2010.]

3.1 Rakenteet

Esittelen ensimmäisenä keskitetyt vertaisverkot. Ne muistuttavat kaikkein eniten asiakas-palvelin-mallia. Erona on kuitenkin se, että keskitetyssä vertaisverkossa tietty palvelin pitää vain yllä tietoa siitä, missä haluttu materiaali sijaitsee ja palauttaa sen käyttäjälle [Liu and Antonopoloulos, 2009]. Se toimii ikään kuin liikenteenohjaajana vertaisverkossa. Vu ja muut [2010] mainitsevat hyvinä puolina kyseisessä rakenteessa nopean tiedonhaun, sillä käyttäjä saa palvelimelta heti tiedon siitä, missä hänen haluamansa tieto sijaitsee. Huonona puolena he esittelevät arkkitehtuurin riippuvuuden palvelimista, sillä jos palvelin kaatuu esimerkiksi ruuhkautumisen takia, ruuhkautuvat mahdolliset muut palvelimet ja palvelu hidastuu tai jopa kaatuu kokonaan. Tunnetuimmat esimerkit keskitetyistä vertaisverkoista ovat Napster-musiikkipalvelu, joka toi tiedostojen jaon tavallisten internet-käyttäjien keskuuteen, sekä tietysti BitTorrenttia hyödyntävät palvelut [Liu and Antonopoloulos, 2009].

Keskittämättömissä vertaisverkoissa ei käytetä erillisiä palvelimia, vaan kaikki käyttäjät ovat samanarvoisia [Vu et al., 2010]. Keskittämättömät vertaisverkot eivät myöskään ole riippuvaisia mistään tietystä solmusta verkossa, kuten keskitetyt vertaisverkot ovat palvelimista. Vun ja muiden [2010] mukaan tämä parantaa huomattavasti vertaisverkkojen luotettavuutta, suorituskykyä ja skaalautuvuutta. Keskittämättömien vertaisverkkojen rakenne voi olla tasainen (flat) tai hierarkkinen (hierachical). Tasaisissa verkoissa toiminnallisuus ja kuorma jaetaan tasaisesti jokaisen käyttäjän kesken, eikä minkäänlaisia eroavaisuuksia tehdä. Useimmat keskittämättömät vertaisverkot ovat tasaisia. Hierarkkisessa verkossa käyttäjät jaetaan eri kerroksiin, esimerkiksi valtiollisella tasolla heidät jaetaan ensin osavaltioihin, sitten osavaltion sisäisiin yliopistoihin ja yliopistojen sisällä vielä eri asuntoloihin. Esimerkkeinä keskittämättömistä vertaisverkoista mainittakoon alkuperäinen Gnutella-tiedostonjakopalvelu ja Freenet, joka on anonymiteettiä korostava elektronisten dokumenttien tallennus- ja jakopalvelu. [Vu et al., 2010.]

Hybridit verkot yhdistelevät keskitettyjen ja keskittämättömien verkkojen parhaita puolia. Verkosta valitaan supersolmuja (super peers), jotka ovat voimakkaampia (suurempi laskentanopeus ja nopeampi internet-yhteys) kuin muut ja pitävät yllä tietoa materiaalien sijainnista [Vu et al., 2010]. Sovelluksista esimerkiksi Kazaa ja Gnutella2 käyttävät hybridiverkkoa [Liu and Antonopoloulos, 2009].

3.2 Etsintämenetelmät

Arkkitehtuuriin vaikuttaa myös käytetty etsintämenetelmä. Rakenteettomassa vertaisverkossa ei pidetä millään tavoin yllä sitä, missä tietty tieto sijaitsee [Liu and Antonopoloulos, 2009]. Solmut pitävät vain huolen omasta materiaalistaan ja tietävät vain muutaman naapurin osoitteet, joihin voi ohjata kyselyjä eteenpäin. Tällaisissa verkoissa tiedon löytäminen voi olla todella hidasta, koska ei ole mitään tietoa siitä, missä se sijaitsee. Sisältöä voi jopa jäädä löytämättä, mikäli ei käydä koko verkkoa läpi. Vastausajastakaan ei ole tietoa, paitsi tietysti huonoimmasta tapauksesta, jossa käydään jokaisessa solmussa. Tietoa voidaan hakea rakenteettomassa verkossa halutulla tavalla. Tässä täytyy huomioida, haluaako nopean, mutta paljon internet-kaistaa kuluttavan vai hitaan, mutta vähemmän kaistaa vievän vaihtoehdon. Esimerkki nopeasta ja kaistaa vievästä tavasta on, että käydään joka askeleella eksponentiaalisen verran solmuja läpi verrattuna edelliseen askeleeseen. [Vu et al., 2010.]

Rakenteellisessa menetelmässä tieto indeksoidaan jollain tavalla. Yleensä tähän käytetään hajautettuja tiivisteitä (distributed hash tables, DHT). Niissä pidetään yllä tietoa siitä, missä mikäkin tieto sijaitsee avainparien avulla. Tietoa jaetaan solmuille, jotta se on nopeasti saatavissa kaikkialta. Näin tuloksena on nopea ja tehokas tiedonhaku. Toisaalta tällaisen tiedon ylläpitämisellä on myös varjopuolensa. Sitä on vaikea ja raskas ylläpitää, etenkin dynaamisissa vertaisverkoissa, joissa käyttäjät voivat poistua ja liittyä milloin haluavat. [Vu et al., 2010.]

Vu ja muut [2010] ovat esitelleet etsintämenetelmät vain keskittämättömien vertaisverkkojen yhteydessä. Liu ja Antonopoloulos [2009] esittävät ne kuitenkin omana kokonaisuutenaan eivätkä tee minkäänlaista yhteyttä näiden kahden välille. Molemmat tavat ovat sopivia, sillä toki ne pohjautuvat keskittämättömiin verkkoihin, mutta pätevät luonnollisesti myös hybridiverkkoihin.

4. Tietoturva ja anonymiteetti

Vertaisverkkojen tietoturva on hyvin vaikea ongelma, ja se on varmastikin yksi syy, jonka takia ne eivät ole vielä yleistyneet kaupallisissa tuotteissa. Tiedostojen tai omien resurssien jakamisessa tuntemattomien kanssa on tottakai olemassa riskinsä. On paljon vaikeampaa ylläpitää tietoturvaa vertaisverkossa kuin käytettäessä asiakas-palvelin-mallia [Vu et al., 2010]. Usein tietoturva saattaakin olla vain luottamuksen varassa. Kun mukaan tuodaan vielä anonymiteetti, tekee se tietoturvallisuusratkaisuista vieläkin vaikeampia. Seuraavaksi esittelen

hieman yleisimpiä hyökkäysmenetelmiä ja niiden jälkeen keinoja saavuttaa anonyymius.

4.1 Hyökkäykset ja niiden torjunta

DoS:ssa (Denial of Service) eli palvelunestohyökkäyksessä verkkoon lähetetään paketteja, jotka estävät muun tietoliikenteen. DDoS (Distributed Denial of Service) eli hajautettu palvelunestohyökkäys on puolestaan sama kuin DoS, mutta hyökkääjällä on käytettävissään monta eri solmua, jotka lähettävät paketteja. Näin ollen hyökkääjää on myös vaikeampi jäljittää, sillä hän ei itse välttämättä ole hyökkääjien joukossa, vaan vain hallitsee hyökkääviä solmuja. Palvelunestohyökkäys voidaan siis helposti sekoittaa korkeaan verkon kuormitukseen, mikä tekee siitä vaikeasti havaittavan. Niiden torjumiseen käytetään pricing-menetelmää. Halutessaan vastauksen pyyntöön solmu joutuu ensin ratkaisemaan jonkin laskennallisen tehtävän, joka hidastaa vasteaikaa siten, että palvelunestohyökkäys hidastuu, eikä välttämättä vaikuta verkkoon millään tavalla. [Engle and Khan, 2006.]

Mies välissä -hyökkäyksessä (Man in the Middle, MitM) hyökkääjä asettaa itsensä kahden eri solmun väliin ja kuuntelee heidän välistä liikennettä. Hän pystyy myös esimerkiksi muuttamaan lähetettyjä viestejä tai lähettämään omia toisen solmun nimissä. Tällaiset hyökkäykset voidaan estää salaamalla viestit ja liittämällä niihin digitaaliset allekirjoitukset. [Engle and Khan, 2006.]

Madot (worms) ovat erittäin vaarallisia vertaisverkoissa, sillä ne leviävät niissä hyvin nopeasti. Normaalisti niiden täytyisi etsiä käyttäjiä, jotka käyttävät tiettyä ohjelmaa, mutta monissa vertaisverkossa ne tietävät varmasti käyttäjällä olevan käytössä sovellus, jolla hän käyttää vertaisverkkoa. Matojen avulla vertaisverkosta voi tehdä vaikkapa bottiverkon, jolla voi suorittaa esimerkiksi DDoS-hyökkäyksiä. Varmin tapa estää matojen leviäminen on tehdä käytettävistä sovelluksista mahdollisimman haavoittumattomia. Toinen keino on antaa käyttäjien tehdä omia sovelluksia vertaisverkon käyttämiseen, jolloin käytettävät ohjelmat ovat monipuolisempia, eikä mato välttämättä tartuta niin montaa käyttäjää. [Engle and Khan, 2006.]

Reitityshyökkäyksissä (routing) hyökkääjä ohjaa muita solmuja vääriin sijainteihin tai solmuille, jotka ovat hyvin epävarmoja tai hitaita. Reitityshyökkäyksessä voidaan myös päivittää muille virheellisiä sijaintitietoja, jolloin ne ohjaavat hyökkääjän haluamiin sijainteihin. Tällaiset hyökkäykset voidaan estää varmistamalla, että solmuun saadaan yhteys ennen kuin sen tietoja päivitetään tai käyttämällä verkossa mainejärjestelmää välttämään yhteyksiä epäluotettaviin solmuihin. [Vu et al., 2010.] Mainejärjestelmissä solmut järjestetään hie-

rarkkisesti. Hierarkia muodostetaan yleisimmin käyttäjien antaman palautteen mukaan. [Yan, 2009.]

Yksi hyökkäyksen muoto on myös verkon hyväksikäyttö. Siinä vertaisverkon käyttäjä ei anna omia resurssejaan muiden käyttöön, mutta hyödyntää silti muiden tarjoamia resursseja. Vaikka tällainen käytös on hyvinkin yleistä, sitä yritetään harvoin ratkaista. Esimerkiksi jotkut BitTorrent-verkot ovat ratkaisseet sen niin, että mitä enemmän käyttäjä jakaa paketteja muille, sitä enemmän se saa ladata muilta. [Engle and Khan, 2006.]

4.2 Identifioinnin estäminen

Anonyymit vertaisverkot jaetaan yleensä korkealatenssisiin ja matalalatenssisiin verkkoihin. Korkealatenssisilla verkoilla on parempi tietoturva, mutta ne voivat olla todella hitaita juuri sen takia. Matalalatenssiset verkot taas omaavat huonomman tietoturvan, mutta toimivat sujuvammin ja ne soveltuvat monipuolisempiin käyttötarkoituksiin. [Wang et al., 2010.] Manzanares-Lopezin ja kumppaneiden [2010] mukaan anonyymiys on kätevä ominaisuus esimerkiksi arkaluontoisten sairauksien käsittelyssä, ilmiannossa tai vaikkapa elektronisessa äänestämisessä. Anonyymit verkot voivat luoda kuitenkin myös eettisiä ongelmia. Esimerkiksi lapsipornon levitys ja kulutus on helpompaa anonyymien vertaisverkkojen kautta.

Lähteen uudelleenkirjoitus (source-rewriting) on järjestelmä, jolla voidaan tehdä vertaisverkosta anonyymi. Sen perusideana on valita joukko solmuja, joiden kautta viesti välitetään, ja jokainen läpikäyty solmu ylikirjoittaa lähettäjän tietoihin omat tietonsa. Tällaisia järjestelmiä ovat muun muassa sekoitukset (mixes), sipulireititykset (onion routing) ja ryhmäykset (crowds).

Ensimmäinen tällaisen järjestelmän toteutus on sekoitus. Siinä jokainen solmu tietää vain edeltäjänsä ja seuraajan. Solmu odottaa, että saa tietyn määrän paketteja itselleen, dekryptaa eli purkaa niistä salauksen, kasaa, järjestelee, poistaa niistä lähettäjän nimen ja tunnistetiedot ja lähettää ne seuraavalle. Esimerkiksi Mixmaster ja GNUnet ovat sekoitusta hyödyntäviä verkkoja. [Manzanares-Lopez et al., 2010.]

Sipulireitityksessä valitut solmut dekryptaavat yhden kerroksen viestin salauksesta ja lähettävät sen seuraavalle. Viimeinen solmu poistaa alimman salauskerroksen ja lähettää viestin vastaanottajalle. [Manzanares-Lopez et al., 2010.] Tor on yksi tunnettu esimerkki tällaisesta verkosta; sillä on satoja tuhansia käyttäjiä päivittäin [Loesing et al., 2010]. Kehittyneempiä versioita ovat muun muassa NISAN ja Torsk [Wang et al., 2010].

Ryhmäyksessä on joukko käyttäjiä, joita sanotaan ryhmäksi. Nämä kaikki haluavat asioida tietyn palvelimen kanssa paljastamatta itseään. Kun ryhmän yksi solmu haluaa ottaa yhteyden palvelimeen, se ottaa yhteyden toiseen solmuun, joka arpoo, kierrättääkö viestin vielä uuden solmun kautta vai välittääkö viestin palvelimelle. Tämä jatkuu niin kauan kunnes palvelimeen otetaan yhteys. Esimerkiksi AP3 on tällainen anonyymi protokolla. [Manzanares-Lopez et al., 2010.]

Levitys järjestelmissä (broadcast systems) ei reititetä viestejä muiden solmujen kautta, vaan niissä salataan viestit vastaanottajan julkisella avaimella. Näin ollen vain halutut vastaanottajat voivat tulkita viestin sisällön. Silloin kun oikeaa sisältöä ei lähetetä, verkossa lähetetään merkityksettömiä viestejä, jotta mahdolliset salakuuntelijat eivät tietäisi mitkä olisivat tärkeitä viestejä. P^5 (Peer-to-Peer Personal Privacy Protocol) ja Hordes ovat esimerkkejä tällaisista systeemeistä. [Manzanares-Lopez et al., 2010.]

5. Sovellukset

Seuraavaksi esittelen vertaisverkkojen avulla toteutettuja sovelluksia. Jokaisen kategorian kohdalla olen pyrkinyt avaamaan toimintamallin vähintään yhdellä esimerkillä, jotta toiminnallisuus tulisi paremmin esille.

5.1 Musiikki- ja elokuvapalvelut

Musiikki- ja elokuvamarkkinat ovat nykyään todella suuria markkina-alueita. Vielä muutamia vuosia sitten kyseisen alan palvelut toimivat lähinnä ”tilaa tuote, saat sen postissa” -periaatteella. Nyt on kuitenkin alkanut ilmaantua palveluita, joita voi käyttää oman koneen ääreltä ja tuotteet saa heti käytettäväksi.

Spotify on vuonna 2008 julkaistu musiikin suoratoistopalvelu, joka toimii vertaisverkkoperiaatteella. Tosin Spotifyn älypuhelinversio ei toimi vertaisverkoissa. Palvelussa käytetään hyödyksi rakenteetonta verkkoa, jossa kappaleet säilytetään käyttäjien kovalevyille varatuissa tilapäismuisteissa. Kun käyttäjä haluaa kuunnella kappaleen, etsii palvelu toisen käyttäjän, jolla kyseinen kappale on. Kappale voi tietysti olla jo valmiiksikin käyttäjän omassa tilapäismuistissa, jolloin yhteyksiä muihin ei tarvitse suorittaa. [Kreitz and Niemelä, 2010.] Spotifyn asetuksista voi itse määrittää, kuinka paljon haluaa antaa kovalevystään palvelun käytettäväksi. Asetuksista voi asettaa tilapäismuistiksi joko 1GB – 100GB tai maksimissaan kymmenen prosenttia vapaasta kovalevyn tilasta. Spotifyn erottaa muista samankaltaisista palveluista sen erillinen sovellus, kilpailijat kun ovat hyvin pitkälti selaimessa pyöriviä, sekä vertaisverkon hyödyntä-

minen [Kreitz and Niemelä, 2010]. Nämä ovat varmasti olleet suuria ominaisuuksia nostamaan palvelusta niin suositun kuin se tällä hetkellä on.

Voddler on Spotifyn ohella toinen menestyksekäs ruotsalainen vertaisverkkoja hyödyntävä multimediapalvelu. Voddler kuitenkin tarjoaa musiikin sijasta elokuvia. Palvelu hyödyntää keskitettyjä resursseja paikantaakseen suoratoistoa tarvitsevan asiakkaan lähimmän vertaisen, jolta kyseinen elokuva löytyy tilapäismuististaan ja ohjaa pyynnön sinne. Itse tiedostojen jako siis tapahtuu keskittämättömästi. [Voddler, 2011.] Elokuvien katsomisen suoratoistona on mahdollistanut internetkaistojen kasvaminen. Mainittakoon kyseisistä palveluista myös Coolstreaming, joka oli yksi ensimmäisistä laatuaan. [Munõz-Gea et al., 2009.]

5.2 Tiedostojenjako ja -levitys

Tiedostojenjako- ja tiedostojenlevityssovellusten katsotaan olevan Naptserin seuraajia [Rodrigues and Druschel, 2010]. Napsterissa käyttäjät valitsevat tiedostot, jotka haluavat antaa muiden saataville. Kun palveluun suorittaa haun, antaa palvelin tiedon siitä, missä haluttu tiedosto on. Tämän jälkeen käyttäjä lataa tiedoston suoraan toisen käyttäjän kovalevyltä. [Shirky, 2001; Rodrigues and Druschel, 2010.] Esimerkki tiedostojenjakoverkoista on Gnutella, jossa kaikki käyttäjät ovat samassa verkossa. Käyttäjät voivat hakea verkosta hakutoiminnolla sisältöä ja ladata kaikkia tiedostoja. Tiedostojenlevityksessä, kuten esimerkiksi BitTorrent-yhteyskäytännössä, käyttäjät etsivät ensin verkosta halutun tiedoston ja muodostavat sen jälkeen, esimerkiksi torrent-tiedoston avulla, vertaisverkon vain niiden käyttäjien kanssa, joilla haluttu tiedosto on jaossa. [Rodrigues and Druschel, 2010.] Vertaisverkot ovat todella kätevä kanava jakaa suuria tiedostoja. Tämän takia esimerkiksi Linux-levityspaketit jaetaan usein BitTorrentin avulla. Kuten kuitenkin jo aiemmin mainitsin, velloo BitTorrenttien yhteydessä usein termi ”piratismi”.

5.3 Pilvilaskenta

Tunnetuin pilvilaskentaa hyödyntävä projekti on SETI@home. Siinä radioteleskooppi etsii avaruudesta maan ulkopuolista elämää radiosignaalien avulla. Teleskooppi vastaanottaa valtavan määrän tietoa, joka lähetetään vapaaehtoisille, jotka ovat asentaneet vaaditun sovelluksen Windows- tai Mac-käyttöjärjestelmälleen. Sovellus toimii vain silloin, kun käyttäjä itse ei tarvitse konettaan, toisin sanoen sovellus korvaa käyttäjän koneen näytönsäästäjän. Sovellus vastaanottaa lähetetyn tiedon ja etsii siitä haluttuja radiosignaaleja. Kun kaikki tieto on käyty läpi, lähettää sovellus tulokset takaisin SETI@homen järjestelmään. [Kor-

pela et al., 2011; Anderson, 2001.] SETI@home pystyy tällä hetkellä yli 550 TeraFLOPS:n (floating-point operations per second) laskentanopeuteen [BOINC-stats, 2011].

5.4 Suoratoisto

Vertaisverkkoja hyödyntävässä suoratoistossa vertaiset lähettävät jatkuvasti heidän saatavilla olevaa sisältöä eteenpäin verkon muille käyttäjille. Mitä enemmän siis on käyttäjiä verkossa, sitä enemmän on sisältöäkin jaossa. [Magharei et al., 2007.] Suoratoistossa on tärkeää juuri pakettien nopea lataaminen, joten siksi vertaisverkot ovat omiaan kyseiseen tarkoitukseen [Rodrigues and Druschel, 2010]. Suoratoistossa käyttäjistä voidaan muodostaa vertaisverkko joko IP:n (Internet Protocol) avulla tai sovelluksen kautta. IP:n avulla tapahtuvassa suoratoistossa suorituskyky on parempi, mutta vastaavasti se kuluttaa enemmän internet-kaistaa. Sovelluksen avulla tapahtuva suoratoisto on myös huomattavasti helpompi toteuttaa. [Kolberg, 2009.] IP:tä hyödyntävissä ratkaisuissakin on olemassa vielä kaksi vaihtoehtoa: puu- tai verkkorakenteisiin pohjautuvat. Näistä kahdesta verkkorakenteisiin pohjautuva on kuitenkin kaistaystävällisempi. [Magharei et al., 2007.]

Useissa suoratoistosovelluksissa, kuten todella suositussa PPLivessä, suositut lähetykset toimivat sulavasti, kun taas vähemmän suositut eivät niinkään. PPLivessä pieni määrä palvelimia lähettää videosta paketteja käyttäjille, jotka lähettävät niitä taas eteenpäin muille käyttäjille ja vastaavasti ottavat vastaan paketteja joita heillä ei vielä ole. [Li et al., 2011.]

5.5 Yhteydenpito

Yhteydenpidossa varmasti tunnetuin vertaisverkkoja hyödyntävä esimerkki on Skype. Palvelun avulla voi soittaa normaaleja puheluita tai videopuheluita muille Skypen käyttäjille suoraan vertaisverkossa [Skype, 2011]. Verkossa käytetään keskitettyä rakennetta muodostamaan yhteys kahden solmun välille, mutta sen jälkeen viestintä tapahtuu suoraan solmujen välillä. Puhelut välitetään kryptattuina. Juuri vertaisverkon hyödyntämisen takia Skype pystyy keskittämään kehityksensä uusiin ominaisuuksiin, eikä palvelun tarvitse ylläpitää suuria palvelinhalleja puhelujen välitykseen. [Skype, 2011.]

Pichat hyödyntää myös vertaisverkkoja, mutta se on kuitenkin pikaviestintäpalvelu. Käyttäjät voivat luoda oman palvelimensa omalle koneelleen ja kutsua haluamansa käyttäjät yhdistämään siihen. Palveluun voi yhdistää myös selaimen kautta. [Pichat, 2011.]

5.6 Hakukoneet

On olemassa myös hakukoneita, jotka hyödyntävät vertaisverkkoja. Doulkeridis ja muut [2009] mainitsevat tällaisten vertaisverkkojen hyödyiksi muun muassa kattavuuden, skaalautuvuuden, pienet kustannukset ja sen, että niillä voidaan välttää monopoli-ilmiö. Esimerkiksi Yacy-hakukoneessa käytetään hajautettuja tiivisteitä indeksoimaan tietoa hakutuloksista käyttäjien omille koneille. Kun käyttäjä suorittaa haun, saa hän vastaukset muilta käyttäjiltä. Palvelussa ei siis tarvita erikseen tietokantoja tallentamaan indeksointitietoja, kuten suurilla hakukonepalveluilla. Yacyn tavoitteena on myös täysin vapaa haku, jossa kukaan ei voi valvoa, mitä hakuja kukin käyttäjä suorittaa. [Yacy, 2011.] Se toimii kontrastina hakukonejäteille, joiden motiivit usein ovat pelkässä rahanteossa, ja näin ollen niiden toiminta ei välttämättä ole kovin pyyteetöntä, mikä laskee käyttäjien luottamusta yksityisyyteen.

6. Käyttömahdollisuuksia ja pohdintaa

Nykyiset vertaisverkkototeutukset ovat mielestäni vielä hyvinkin alkukantaisia. Jos mietimme esimerkiksi BitTorrent-toteutusta, on siinä haettava ensin verkosta haluttua tiedostoa vastaava torrent-tiedosto ja sen jälkeen käytettävä erillistä sovellusta itse tiedoston lataukseen ja jakamiseen. Toki on olemassa toteutuksia, kuten BitTorrent-lisäosia selaimiin, jotka hieman rikkovat kaavaa, mutta pääasiassa käyttö on edellen melko alkeellista. Vertaisverkkoja ei myöskään hyödynnetä kovin paljoa mobiilimaailmassa, johon se tulee varmasti leviämään. Vertaisverkkoja hyödyntävät sovellukset ja palvelut ovatkin olleet enimmäkseen vielä erilaisten voittoa tavoittelemattomien yhteisöjen tai yksilöiden aikaansaannoksia. Kuitenkin tekniikoiden kehittyessä ja potentiaalin tullessa kaikille selväksi tulevat vertaisverkot varmasti yleistymään räjähdysmäisesti myös kaupallisella puolella ja näemme enemmän Spotifyn, Voddlerin ja Skypen kaltaisia käytännöllisiä sovelluksia.

Miljoonat tietokoneet ovat päällä yötäpäivää ympäri maailmaa, joten SETI@homen kaltaisia palveluita voitaisiin hyödyntää todella paljon enemmän, kuten toteavat myös Doulkeridis ja muut [2009]. Esimerkiksi suoratoistossa, jossa kaistaa ei voi ikinä olla liikaa, voitaisiin hyödyntää palvelua käyttävien asiakkaiden koneiden joutoaikaa. Itse ainakin olisin valmis tällaiseen, jos vain saisin vastineeksi hyvällä kuvanlaadulla toimivan suoratoiston silloin, kun itse sellaista tarvitsen. Toinen esimerkki voisi olla koneiden joutoajan hyödyntäminen verkossa pelattavien moninpelien palvelinten korvaamisessa. Käyttäjät,

jotka eivät sillä hetkellä ole koneensa ääressä, voisivat lahjoittaa koneensa ja internet-kaistansa muiden pelaajien hyödyksi. Tällaiselle toiminnalle voisi asettaa jonkinnäköisen porkkanan, jossa vaikkapa aktiivinen lahjoittaja saisi itselleen ilmaiseksi etuisuuksia, kuten esimerkiksi ilmaista peliaikaa.

Doulkeridis ja muut [2009] mainitsevat eräänä kiinnostusta herättäneenä tutkimusalueena vertaiverkkoja käyttävät sosiaaliset verkostot. Niiden hyötynä olisi se, että kaikki tieto ei olisi vain yhden suuren yhtiön tietokannoissa, vaan jokainen voisi hallita paikallisesti omaa informaatiota itsestään ja jopa käyttää palvelua ilman internet-yhteyttä [PeerSoN, 2011]. Tällainen käytäntö olisi varmasti omiaan haastamaan yksityisyysasetusten kanssa vaikeroivan Facebookin. Eräs tällainen kehitteillä oleva palvelu on PeerSoN, josta on kehitetty jo jonkinasteinen prototyyppi [PeerSoN, 2011].

Kuten aiemmin jo mainitsin, vertaisverkkoja ei ole hyödynnetty älypuhelinsovelluksissa vielä kovinkaan paljoa. Mobiiliverkkojen lataus- ja jakonopeudet ovat vielä todella hitaita ja vertaisverkkojen aiheuttama liikenne hidastaisi niitä vielä entisestään. Nykyään kuitenkin jo lähes jokainen kaupasta ostettu kännykkä toimii langattomassa lähiverkossa, joiden kautta jakonopeudet ovat yleensä huomattavasti korkeammat. Esimerkiksi oman sijaintitiedon jakaminen vertaisverkon kautta olisi taas yksi askel yksityisempään ja rajatumpaan tiedonjakoon päin.

Liun ja muiden [2009] mukaan laajoissa vertaisverkoissa keskeisimpänä haasteena on sisällön löytäminen tehokkaasti. Sisällön löytäminen on varmasti kehittynyt hybridiverkkojen myötä, mutta siinä on siltikin vielä paljon parannettavaa. Yleisestikin vertaisverkkojen arkkitehtuurit olisi hyvä tutkia ja tunnistaa jo tässä vaiheessa paremmin, jotta kehitystä osattaisiin viedä tehokkaammin oikeaan suuntaan ja käyttää hyväksi oikeita toteutuksia oikeissa paikoissa.

Viiteluettelo

[Anderson, 2001] David Anderson, SETI@home. In: Andy Oram (ed.), *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*. O'Reilly & Associates, 2001, 67–76.

[BOINCstats, 2011] BOINCstats, SETI@Home. http://boincstats.com/stats/project_graph.php?pr=sah. Checked 9.12.2011.

[Doulkeridis et al., 2009] Christos Doulkeridis, Akrivi Vlachou, Kjetil Nørvåg and Michalis Vazirgiannis, Distributed semantic overlay networks. In: Xue-

min Shen, Heather Yu, John Buford and Mursalin Akon (eds.), *Handbook of Peer-to-Peer Networking*. Springer, 2009, 463–494.

[Engle and Khan, 2006] Marling Engle and Javed I. Khan, Vulnerabilities of P2P Systems and a Critical Look at their Solutions. Kent State University, Computer Science Dept, Technical Report 2006-11-01, November 2006. Also available as *http://www.medianet.kent.edu/techreports/TR2006-11-01-p2pvuln-EK.pdf.*

[Kolberg, 2009] Mario Kolberg, Employing multicast in p2p overlay networks. In: Xuemin Shen, Heather Yu, John Buford and Mursalin Akon (eds.), *Handbook of Peer-to-Peer Networking*. Springer, 2009, 861–874.

[Korpela et al., 2011] Eric J. Korpela, Jeff Cobb, Matt Lebofsky, Andrew Siemion, Joshua Von Korff, Robert C. Bankay, Dan Werthimer and David Anderson, Candidate identification and interference removal in SETI@home. In: Douglas A. Vakoch (ed.), *Communication with Extraterrestial Intelligence.* State University of New York Press, 2011, 37-44.

[Kreitz and Niemelä, 2010] Gunnar Kreitz and Fredrik Niemelä, Sporify – large scale, low latency, p2p music-on-demand streaming. In: *Proc. of 10th IEEE P2P'10*. 1–10.

[Li et al., 2011] Ruixuan Li, Guoqiang Gao, Weijun Xiao, Zhiyong Xu, Measurement study on PPLive based on channel popularity. In: *Proc. of 9th CNSR 2011.* 18–25.

[Liu and Antonopoloulos, 2009] Lu Liu and Nick Antonopoloulos, From client-server to p2p networking. In: Xuemin Shen, Heather Yu, John Buford and Mursalin Akon (eds.), *Handbook of Peer-to-Peer Networking.* Springer, 2009, 71–90.

[Liu et al., 2009] Lu Liu, Jie Xu, Duncan Russell and Zongyang Luo, Self-adaptation and self-organization for search in social-like peer-to-peer networks. In: Heather Yu, John Buford and Mursalin Akon (eds.), *Handbook of Peer-to-Peer Networking.* Springer, 2009, 495–530.

[Loesing et al., 2010] Karsten Loesing, Steven J. Murdoch and Roger Dingledine, A Case Study on Measuring Statistical Data in the Tor Anonymity Network. University of Cambridge, Computer Laboratory, 2010. Also available as *metrics.torproject.org/papers/wecsr10.pdf.*

[Rodrigues and Druschel, 2010] Rodrigo Rodrigues and Peter Druschel, Peer-to-peer systems. *CACM* **53**, 10 (2010), 75–82.

[Magharei et al., 2007] Nazanin Magharei, Reza Rejaie and Yang Guo, Mesh or multiple tree: a comparative study of live p2p streaming approaches. In: *Proc.*

of 26t h IEEE International Conference on Computer Communications INFOCOM 2007. 1424–1432.

[Manzanares-Lopez et al., 2009] Pilar Manzanares-Lopez, Juan Pedro Munõz-Gea, Josemaria Malgosa-Sanahuja and Juan Carlos Sanchez-Aarnoutse, Anonymity in p2p systems. In: Xuemin Shen, Heather Yu, John Buford and Mursalin Akon (eds.), *Handbook of Peer-to-Peer Networking.* Springer, 2009, 785–812.

[Minar and Hedlund, 2001] Nelson Minar and Marc Hedlund, A network of peers: peer-to-peer models through the history of the internet. In: Andy Oram (ed.), *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*. O'Reilly & Associates, 2001, 3–20.

[Munõz-Gea et al., 2009] Juan Pedro Munõz-Gea, Josemaria Malgosa-Sanahuja, Pilar Manzanares-Lopez and Juan Carlos Sanchez-Aarnoutse, Providing VoD streaming using p2p networks. In: Xuemin Shen, Heather Yu, John Buford and Mursalin Akon (eds.), *Handbook of Peer-to-Peer Networking*. Springer, 2009, 1025–1041

[PeerSoN, 2011] PeerSoN, PeerSoN: privacy-precerving p2p social networks. http://www.peerson.net/index.shtml. Checked 13.12.2011.

[Pichat, 2011] Pichat, Pichat chat features. http://www.pichat.net/documentation/pichat_features. Checked 9.12.2011.

[Shirky, 2001] Clay Shirky, Listening to Napster. In: Andy Oram (ed.), *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*. O'Reilly & Associates, 2001, 21–37.

[Skype, 2011] Skype, What Are P2P Communications. https://support.skype.com/en-us/faq/FA10983/What-are-P2P-communications. Checked 5.12.2011.

[Voddler, 2011] Voddler, About Voddler. http://voddlertalk.voddler.com/om/. Checked 9.12.2011.

[Vu et al., 2010] Quang Hieu Vu, Mihai Lupu and Beng Chin Ooi, *Peer-to-Peer Computing. Principles and Applications.* Springer-Verlag, 2010.

[Wang et al., 2010] Qiyan Wang, Prateek Mittal and Nikita Borisov, In search of an anonymous and secure lookup: attacks on structured peer-to-peer anonymous communication systems. In: *Proc. of 17th ACM Conference on Computer and Communications Security*, 308–318.

[Yacy, 2011] Yacy, Web Search by the people, for the people. http://yacy.net/en/index.html. Checked 9.12.2011.

[Yan, 2009] Lu Yan, A formal architectural model for peer-to-peer systems. In: Xuemin Shen, Heather Yu, John Buford and Mursalin Akon (eds.), *Handbook of Peer-to-Peer Networking*. Springer, 2009, 1295-1314.

Sovelluskomponenttien välisten viestien tietoturvariskit Android-käyttöjärjestelmässä

Ville Saarinen

Tiivistelmä.

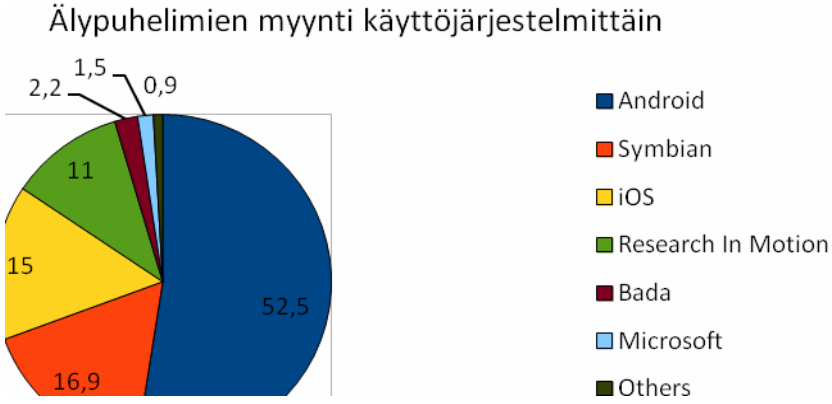
Älypuhelimien ja muiden mobiililaitteiden yleistyessä niistä on tullut mielenkiinnon kohde myös haitallisten sovellusten kehittäjille. Tässä tutkielmassa käsitellään Android-käyttöjärjestelmää ja sille tehtyjen sovellusten haavoittuvuuksia ja niiden ehkäisyä.

Avainsanat ja -sanonnat: Android, Intent, tietoturva

CR-luokat: D.4.6

1. Johdanto

Android on mobiililaitteille suunnattu avoimen lähdekoodin käyttöjärjestelmä, jonka Google julkaisi vuonna 2007. Ensimmäinen Androidia käyttänyt älypuhelin tuli markkinoille vuonna 2008, kun G1-puhelin julkaistiin. [Enck *et al.*, 2009] Vuoden 2011 kolmannella neljänneksellä Androidin markkinaosuus oli jo 52,5 % kaikista myydyistä älypuhelimista, kuten kuvasta 1 voidaan nähdä. [Gartner, 2011]



Kuva 1: Vuoden 2011 kolmannella neljänneksellä myytiin yli 115 000 000 älypuhelinta, joista yli 50 % oli Android-puhelimia. [Gartner, 2011]

Kuten mobiilikäyttö-järjestelmillä yleensä, myös Androidilla on oma kauppa – Android Market, josta käyttäjät voivat ladata kolmansien osapuolten tekemiä sovelluksia mobiililaitteisiinsa. Nämä sovellukset voivat kuitenkin muodostua tietoturvariskiksi, jotka asettavat käyttäjien yksityiset tiedot vaaraan. Tässä tutkielmassa on käsitelty erilaisia tapoja, joilla haittasovellukset voivat käyttää hyväkseen laillisten Android-sovellusten sisältämiä virheitä ja miten näitä tieto-

turvariskejä voidaan vähentää sovellusten oikeanlaisella toteutuksella. Tutkielmassa on keskitytty sovelluskomponenttien välisiin viesteihin ja niiden käyttöön hyökkäyksissä Android-mobiililaitteita ja niitä käyttävien ihmisten yksityisiä tietoja kohtaan.

2. Android-sovellus

Tässä luvussa esitellään sovelluksen komponentit ja niiden välinen viestintä yksinkertaisen herätyskellosovelluksen avulla.

2.1. Komponentit

Sovellukset koostuvat komponenteista, joita on neljää erilaista tyyppiä. Komponenttityypit ovat:

- *Activity.* Sovelluksen käyttöliittymä. Ainoastaan yhdellä Activityllä voi olla prosessointi- ja näppäimistöfokus kerrallaan, muut Activityt keskeytetään.
- *Service.* Taustalla pyörivä komponentti, joka mahdollistaa esimerkiksi musiikin soittamisen myös sen jälkeen, kun ko. sovelluksen käyttöliittymä on suljettu.
- *BroadcastReceiver.* Komponentien lähettämien viestien vastaanottaja. Voidaan käyttää esimerkiksi Service- ja Activity-komponenttien välissä, kuten esimerkkisovelluksessa.
- *ContentProvider.* Säilyttää dataa ja jakavaa sitä muille komponenteille. Android API sisältää useita natiiveja ContentProvidereita, kuten esimerkkisovelluksessa käytetyn MediaStoren, jonka avulla päästään käsiksi laitteen muistissa oleviin kuva-, ääni- ja videotiedostoihin.

2.2. Komponenttien välinen viestintä

Komponenttien välistä viestintää tarvitaan yksinkertaisimmissakin sovelluksissa. Viesteillä mm. käynnistetään uusia komponentteja ja annetaan niille uutta dataa, jonka varassa ne toimivat. Komponenttien välisessä vuorovaikutuksessa käytetään ns. Intent-objekteja, joita voidaan käyttää joko sovelluksen sisäisessä tai sovellusten välisessä viestinnässä. Intentiin voidaan määritellä sisällöksi vastaanottajan lisäksi seuraavaa sisältöä:

- *Action.* Halutun tai tapahtuneen toiminnan nimi
- *Data.* URI datalle. Esimerkiksi tiedoston osoite muistissa.
- *Extra.* Lisätietoja vastaanottajalle, joka voisi olla esimerkiksi herätyskellosovelluksessa äänenvoimakkuus ja torkkutoiminnon kesto.
- *Type*. Määrittelee tietyn datatyypin, jota käytetään.
- *Category.* Sen komponentin tyyppi, jonka halutaan vastaanottaa ko. Intent.

- *Flag.* Lippumuuttuja, joka määrittelee kuinka Intentiä tulee käsitellä.

Intent voi olla tyypiltään eksplisiittinen tai implisiittinen [Chin *et al.,* 2011]. Eksplisiittinen Intent määrittää tarkasti tietyn vastaanottajakomponentin, kun taas implisiittinen Intentin vastaanottajaa ei määritellä ollenkaan, vaan se voi olla mikä tahansa sovellus, joka tukee haluttua toimintoa. Käytännössä implisiittisen Intentin vastaanottajan lopullinen määritteleminen jää käyttäjälle, joka valitsee hänelle esitetystä listasta haluamansa sovelluksen. Android valitsee ko. listan sovellukset Intent-suodattimilla, joista lisää myöhemmin. Chin ja muut [2011] käyttävät esimerkkinä sovellusta, joka säilyttää yhteystietoja: Kun käyttäjä klikkaa yhteystiedon katuosoitetta, tarvitsee sovelluksen pyytää toista sovellusta näyttämään se kartalla. Yhteystietosovellus voisi joko lähettää eksplisiittisen Intentin suoraan Google Maps -sovellukselle tai vaihtoehtoisesti se voisi lähettää implisiittisen Intentin, joka suunnattaisiin mille tahansa karttapalveluita tarjoavalle sovellukselle (esim. Yahoo! Maps tai Bing Maps), joista käyttäjä voisi valita haluamansa.

Intentejä voidaan siis käyttää käynnistämään uusia Activityjä, mutta niillä on myös muita käyttötarkoituksia. Kuten aiemmin mainitsin, on olemassa BroadcastReceiver-komponentteja, jotka vastaanottavat viestejä. Nämä komponentit voivat kuunnella paitsi sovellusten myös järjestelmän lähettämiä viestejä ja reagoida niihin. Näitä Intentejä kutsutaan broadcasteiksi. Android ilmoittaa järjestelmän broadcasteilla esimerkiksi silloin, kun Internet-yhteyden tilassa on tapahtunut muutos tai laite on kytketty laturiin. Jos Intentillä on monta soveltuvaa BroadcastReceiveriä vastaanottamassa sitä ja vastaanottajien järjestyksellä on merkitystä, voidaan broadcast lähettää niille yksi kerrallaan halutussa järjestyksessä, jolloin puhutaan järjestetystä broadcastista (ordered broadcast). Tässä tapauksessa BroadcastReceiverit pystyvät muuttamaan tarvittaessa Intentin sisältöä. [Meier, 2010] Broadcastit voivat olla tyypiltään myös pysyviä (sticky), jolloin ne lähetetään niin kauan uudestaan, kunnes lähetys lopetetaan erillisellä käskyllä. Intentejä käytetään myös Service-komponenttien käynnistämiseen tai niille uusien tietojen syöttämiseen. Jotta Activity tai Service voi vastaanottaa Intentejä, tarvitsee ne ilmoittaa sovelluksen asetus- eli manifest-tiedostossa. BroadcastReceiverit voidaan rekisteröidä myös ajonaikaisesti.

Jos komponentin halutaan olla julkinen, eli sen halutaan pystyvän vastaanottamaan muiden sovelluksien lähettämiä Intentejä, tulee se määritellä sovelluksen manifestissa. Se, mille komponentille implisiittinen Intent lähetetään tai mitkä sovellukset valitaan käyttäjälle esitettyyn listaan, määräytyvät Intent-suodattimilla. Suodattimet voivat karsia Intentejä niiden sisältämien data-, category- ja action-tiedoilla. Intent-suodattimien käytön kanssa tulee olla kuitenkin tarkkana, sillä Android tulkitsee kaikki ne komponentit julkisiksi, joille on määritelty yksi tai useampi suodatin. Chinin ja muiden [2011] esimerkkitapaus

on komponentti, joka editoi kuvia. Se voisi sisältää Intent-suodattimen, jossa se määrittelee hyväksyvänsä sellaiset Intentit, jotka sisältävät seuraavat tiedot:

- Action = Intent.ACTION_EDIT
- Type = ”image/*”.

Vastaanottajaksi hyväksytään sellaiset komponentit, joiden Intent-suodattimessa on määritelty vähintään ne tietotyypit, jotka ko. Intentissäkin. Intent-suodattimet eivät ole kuitenkaan turvallisuusmekanismi. Jos monella sovelluksella on komponentteja, jotka käsittelevät samantyyppisiä Intentejä, tarvitaan jokin tapa valita yksi niistä. BroadcastReceiverien tapauksessa valinta tapahtuu prioriteettitason mukaan, jolloin korkeimman prioriteetin omaava vastaanottaja saa Intentin. Kuten aiemmin mainitsin, Activityjen kohdalla valinta perustuu käyttäjän päätökseen. Jos useampi Service pystyy käsittelemään Intentin, Android valitsee niistä sattumanvaraisesti yhden. [Chin *et al.*, 2011]

2.3. Sovellusten oikeudet

Android käyttää Javasta tuttua ns. hiekkalaatikkomallia, joissa jokainen sovellus on eristetty toisistaan omalle ”hiekkalaatikolle” ja niille on myönnetty omat, sovelluskohtaiset, oikeudet käyttää laitteen resursseja [Davi *et al.,* 2010]. Esimerkiksi peli voisi käyttää laitteen Internet-yhteyttä ja karttasovellus sisäänrakennettua GPS:ää, mutta kummallakin sovelluksella on oikeudet ainoastaan niihin vaarallisiin toimintoihin, joihin niille on myönnetty lupa asennuksen yhteydessä. Tämä ajattelutapa eroaa perinteisestä, koska perinteisesti oikeudet on yhdistetty sovellusten sijaan käyttäjiin [Porter Felt *et al.,* 2010]. Asennuksen yhteydessä tapahtuvan oikeuksien myöntämisen edut perinteiseen malliin nähden ovat Porter Feltin ja muiden [2010] mukaan seuraavat:

- Käyttäjiä voidaan varoittaa sovelluksen vaarapotentiaalista.
- Haavoittuvien sovellusten haittavaikutukset on rajattu niille myönnettyihin oikeuksiin.
- Tutkimuksissa on helpompi rajata aineistoa, koska voidaan keskittyä niihin sovelluksiin, joilla on vaarallisiksi luokiteltuja oikeuksia.

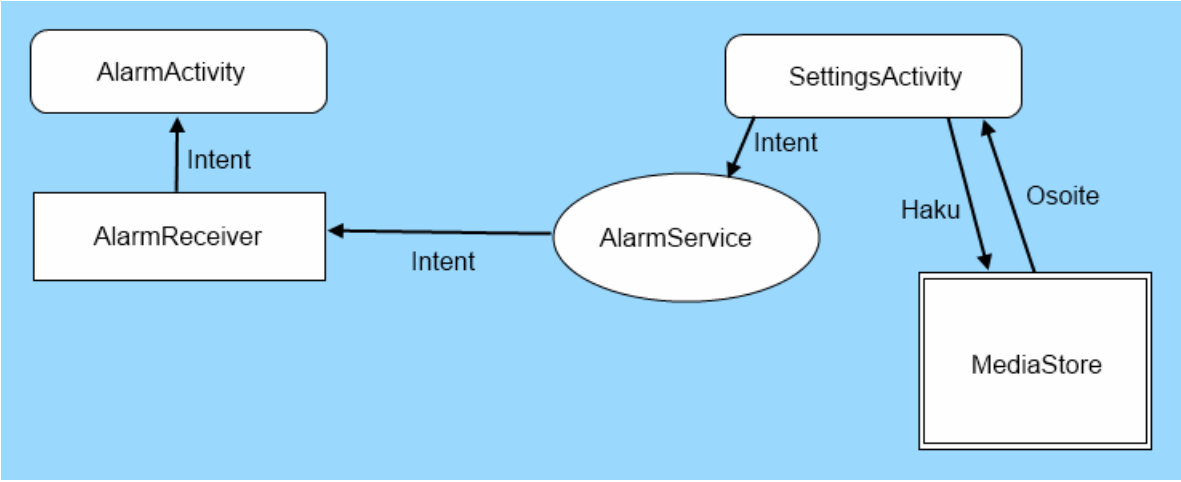
Androidin API 8, eli Android 2.2, sisältää 134 eri oikeutta, jotka on jaettu neljään kategoriaan niiden vaarallisuuden perusteella [Porter Felt *et al.,* 2011]:

- *Normal.* Toiminnot, jotka voivat olla käyttäjän kannalta kiusallisia tai ärsyttäviä, mutta eivät vahingollisia, esimerkiksi oikeus taustakuvan asettamiseen. Käyttäjän ei tarvitse erikseen antaa lupaa näihin toimintoihin.
- *Dangerous.* Vaaralliseksi luokitellut toiminnot, kuten äänen tallentaminen tai kameran käyttäminen.
- *Signature* ja *SignatureOrSystem.* Erittäin vaaralliset toiminnot, kuten käyttäjätietojen poistaminen. Näitä oikeuksia pystyvät käyttämään oikeastaan vain valmiiksi asennetut sovellukset, koska Signature- ja

SignatureOrSystem-oikeuksia käyttäviltä sovelluksilta vaaditaan laitteistovalmistajan sertifikaatti. Jos sertifikaattia ei ole, jätetään pyyntö oikeuksien myöntämisestä huomioimatta. SignatureOrSystem-oikeudet voidaan myöntää myös sovelluksille, jotka ovat asennettu niille varattuihin järjestelmäkansioihin, joihin ei ole muilla ohjelmilla

Oikeuksia voidaan asettaa myös komponenttikohtaisesti sovelluksen manifestitiedostossa tai ohjelman ajon aikana. Näiden oikeuksien tarkoituksena on suojalla komponentteja haitallisilta sovelluksilta. Ohjelmistokehittäjä voi itse luoda omat oikeudet, joita vaaditaan komponenteilta, jotka yrittävät käyttää hänen tekemiään sovelluksen osia. Oikeudet voidaan tarkistaa niin komponenttia käynnistettäessä kuin myös ajon aikana kooditasolla. [Six, 2011]

2.4. Esimerkkisovellus



Kuva 2: Esimerkkinä on yksinkertainen herätyskellosovellus.

Herätyskellosovelluksen toimintaperiaate on melko yksinkertainen ja pelkistetty (kuva 2). Se sisältää kaksi käyttöliittymää, joista toinen on asetusruutu (SettingsActivity) ja toinen herätyksen aikainen näyttö (AlarmActivity). Sovelluksessa on yksi Service- ja yksi BroadcastReceiver-komponentti, joiden lisäksi käytetään MediaStore-ContentProvideria.

Jotta SettingsActivityssä olisi mahdollista valita tietty hälytysääni herätykselle, tarvitaan tieto siitä mitä äänitiedostoja laitteen muistissa on tarjolla ja missä päin muistia ne sijaitsevat. Tässä kohtaa mukaan tulee MediaStore, joka on Android API:n tarjoama ContentProvider. Sen avulla saamme selville laitteen kaikki mediatiedostot ja mm. niiden osoitteet muistissa, joskin tämän sovelluksen tapauksessa on olennaista keskittyä vain äänitiedostoihin. Kun käyttäjä on valinnut haluamansa ajankohdan ja hälytysäänen voidaan hälytys asettaa. Tämä tapahtuu Android API:n natiiviin AlarmServicen avulla. Sille annetaan mm. hälytyksen ajankohta ja Intent, joka tässä tapauksessa on eksplisiittinen ja sisältää käyttäjän valitseman hälytysäänen URI:n (Koodi 1).

AlarmService lähettää valittuna ajankohtana broadcastin AlarmReceiverille,

joka puolestaan välittömästi käynnistää samaisella Intentillä AlarmActivityn (Koodi 2), joka soittaa hälytysäänen ja näyttää käyttäjälle käyttöliittymän, jolla hälytys voidaan sammuttaa.

3. Intentien käyttö hyökkäyksissä

Hyökkäykset ovat erityisesti implisiittisten Intentien ja julkisiksi määriteltyjen komponenttien ongelma, koska ne ovat näkyvillä kaikille muille komponenteille. Jos sovelluksen kehittäjät eivät ole huolehtineet tarpeellisista suojakeinoista, hyökkäykset voivat onnistua, koska Android jättää tämän vastuun heille [Davi *et al.,* 2010]. Chin ja muut [2011] tuovat esille seitsemän erilaista hyökkäystyyppiä, jotka esitellään tässä luvussa.

3.1. Haitallinen vastaanottaja

Haittasovellukset pystyvät kaappaamaan haavoittuvia implisiittisiä Intentejä ja käyttävät niiden tietoja hyväkseen sekä lisäävät niihin omaa dataa muuttaen täten sovelluksen toimintaa.

3.1.1. Broadcast-varas

Julkiset broadcastit ovat haavoittuvia sekä passiivisille salakuuntelijoille että aktiivisille hyökkääjille. Passiiviset salakuuntelijat voivat kaikessa hiljaisuudessa lukea broadcastin sisältöä kenenkään huomaamatta. Tämä tapahtuu siten, että haittasovelluksen BroadcastReceiverin Intent-suodatimeen sisällytetään kaikki action-, data- ja category-kentät, jolloin se voi vastaanottaa kaikkia julkisia broadcasteja. Pysyvät broadcastit ovat erityisen haavoittuvia salakuuntelijoille, koska niiden toimintaikkuna on niin suuri. Tämän lisäksi niitä ei voi suojata oikeuksilla.

Järjestettyjen broadcastien ongelmana on palvelunesto- ja injektiohyökkäykset, joita aktiiviset sovellukset tekevät. Koska nämä broadcastit lähetetään BroadcastReceivereille prioriteetin mukaisessa järjestyksessä, voi haittasovellus asettaa itsensä tämän jonon kärkeen ja estää broadcastin etenemisen muille vastaanottajille. Tämän tyyppisiä hyökkäyksiä kutsutaan palvelunestohyökkäyksiksi. Koska lähettäjäkomponentit saavat BroadcastReceivereiltä paluuviestinä Intentin, haitallinen sovellus voi muuttaa tämän Intentin sisältöä ja täten komponentin toimintaa, jolloin kyseessä on injektiohyökkäys.

3.1.2. Activity-kaappaus

Activity-kaappauksessa haitallinen Activity käynnistetään tarkoitetun Activityn sijasta. Hyökkäyksessä voisi käyttää esimerkiksi sovellusta, joka kerää rahaa lahjoituksin. Kun käyttäjä painaa ”Donate Here” -nappia, aloittaa sovellus implisiittisellä Intentillä uuden Activityn, joka kysyy käyttäjältä maksutiedot. Jos haitallinen Activity kaappaa tämän Intentin, voi hyökkääjä saada käyttäjän

tiedot. [Chin *et al.*, 2011]

Aina tällainen hyökkäys ei kuitenkaan ole mahdollista, koska kuten aiemmin mainitsin, jos useampi Activity pystyy käsittelemään ko. Intentiä, näytetään käyttäjälle lista käytettävissä olevista Activityistä. Tässä tapauksessa käyttäjä voi valita turvalliseksi tietämänsä sovelluksen ja hyökkäys epäonnistuu. Chin ja muut [2011] esittävät kuitenkin tähän kaksi tapaa, joilla hyökkääjä voi hämätä käyttäjää valitsemaan haittasovelluksen laillisen sovelluksen sijaan.

- Haittasovellus voidaan nimetä hämäävästi, jolloin käyttäjä epähuomiossa sen valitsee.
- Haitallinen sovellus voi tarjota hyödyllisiä palveluita, jolloin käyttäjä valitsee sen vapaaehtoisesti. Tällaisessa tapauksessa käyttäjä voisi valita esimerkiksi haitallisen selaimen laitteen oletusselaimeksi, jolloin valintaa ei enää jatkossa joudu edes tekemään.

Vaikka tämän kaltaisen hyökkäyksen onnistumisella on omat haasteensa, onnistuneen hyökkäyksen seuraamukset voivat olla merkittävät.

3.1.3. Service-kaappaus

Haitallisia Servicejä voidaan käynnistää laillisen komponentin sijasta samalla periaatteella kuin Activity-kaappauksissa. Haitallinen Service voi varastaa ja palauttaa väärää dataa sekä valehdella suorittaneensa pyydetyt tehtävät. Service-kaappaus on siinä mielessä kuitenkin erilainen Activity-kaappaukseen nähden, että se ei ole käyttäjälle näkyvä johtuen Service-komponentin luonteesta. Käyttäjä ei pysty valitsemaan haluamaansa Serviceä, koska useamman tehtävään soveltuvan Servicen tapauksessa valinta tehdään Androidin toimesta sattumanvaraisesti.

3.1.4. Oikeuksien kaappaaminen

Aiemmassa luvussa oli asiaa oikeuksista, jotka voidaan antaa sekä manifestissa että Intentien avulla, jos vastaanottajalla niitä ei ole entuudestaan. Intentit voivat sisältää URI-osoitteita, jotka viittaavat ContentProviderin sisältämään dataan. Tämän datan lukemiseen ja muuttamiseen tarvitaan oikeuksia, jotka voidaan Intentin vastaanottajalle myöntää, kunhan ko. ContentProvider sen sallii. Oikeuksien myöntäminen tapahtuu asettamalla Intentin lippumuuttujalle toinen seuraavista arvoista:

- FLAG_GRANT_READ_URI_PERMISSION
- FLAG_GRANT_WRITE_URI_PERMISSION.

Jos haitallinen komponentti anastaa tällaisen Intentin, pystyy se lukemaan tai muuttamaan URI:n osoittamaa dataa.

Samaan tapaan ns. PendingIntentit mahdollistavat oikeuksien joutumisen haitalliselle komponentille. PendingIntent, joka säilyttää kaikki tekijäkompo-

nenttinsa oikeudet, vaikka se lähetettäisiin edelleen kolmannellekin osapuolelle. Tästä ominaisuudesta johtuen tämän tyyppiset Intentit mahdollistavat tekijäkomponenttien oikeuksien väärinkäytön joutuessaan haitallisen komponentin kaappaamaksi.

3.2. Haitallinen Intentin lähettäjä

Haittasovellukset voivat käynnistää julkiseksi määriteltyjä komponentteja ja esiintyä niille laillisina, mikäli sovelluksen kehittäjä ei ole huolehtinut tarvittavista tarkistuksista.

3.2.1. Broadcast-injektiohyökkäys

Jos julkiseksi määritelty BroadcastReceiver luottaa sokeasti vastaanottamiinsa Intenteihin, se voi toimia sopimattomasti tai käyttää saastunutta dataa näistä Intenteistä. Koska BroadcastReceiverit antavat usein käskyjä ja dataa eteenpäin Serviceille ja Activityille, voi haittasovellus toimia läpi ohjelman.

BroadcastReceiverit, jotka rekisteröivät vastaanottamaan järjestelmän lähettämiä Intentejä, ovat erityisessä vaarassa tällaisille injektiohyökkäyksille. Kun komponentti rekisteröi itsensä vastaanottamaan järjestelmän viestejä, tulee siitä samalla julkinen. Jos komponentti ei tarkista Intentin tietoja, voidaan se huijata käyttämään sellaisia toimintoja, joita ainoastaan käyttöjärjestelmän tulisi pystyä käynnistämään. [Chin *et al.*, 2011]

3.2.2. Activityn käynnistäminen

Julkisiksi määriteltyjä Activityjä voidaan käynnistää sekä implisiittisillä että eksplisiittisillä Intenteillä. Activityn käynnistämishyökkäyksiä on kolmea eri tyyppiä [Chin *et al.*, 2011]:

- Activityn käynnistäminen haitallisella Intentillä voi vaikuttaa sovelluksen tilaan tai muokata sen dataa sovelluksen taustalla. Jos Intentiä ei varmisteta oikealla tavalla ja Activity käyttää sen tietoja, voi sovelluksen tietovarastot korruptoitua.
- Käyttäjää voidaan huijata esimerkiksi muuttamaan väärän sovelluksen tietoja. Tämä tapahtuu käynnistämällä eri sovelluksen asetuskomponentti, kun käyttäjä pyrkii muuttamaan haittasovelluksen asetuksia.
- Laillinen Activity voi vuotaa arkaluontoisia tietoja palauttamalla toimintansa tuloksen sen käynnistäneelle komponentille.

3.2.3. Servicen käynnistäminen

Service-komponenttien käynnistämiseen kohdistuvat hyökkäykset ovat samantyylisiä kuin alakohdassa 2.2.2 esitelty hyökkäysmalli, mutta Service-komponenttien luonteesta johtuen niihin kohdistuvat hyökkäyksillä on parempi mahdollisuus onnistua. Tämä johtuu siitä, että Servicet usein käyttävät Intentien

tietoja Activityjä enemmän ja tiedot ovat Service-komponenttien toiminnalle tärkeämpiä. Koska Service-komponenttien sisältämiä metodeja voidaan useimmissa tapauksissa kutsua monta kertaa, on haittasovelluksen toimintaikkuna suurempi.

4. Hyökkäyksiltä puolustautuminen

Kuten aiemmin mainitsin, on komponenttien ja Intentien suojaaminen jätetty ohjelmoijien vastuulle, eikä Android huolehdi niistä. Android suojaa kyllä puhelimen haittasovelluksilta, mutta tarjoaa sovelluksille pahasti rajoittuneen infrastruktuurin suojella itseään [Ongtang *et al.*, 2009]. Ohjelmistokehittäjillä on olemassa kuitenkin keinot tehdä sovelluksista turvallisempia, kunhan he muistavat oikeat toteutustavat. Chin ja muut [2011] luettelevat ohjelmistokehittäjille seuraavia ohjenuoria:

- Lähetettäessä yksityisiä tietoja tulisi käyttää eksplisiittisiä Intentejä kunhan se on vain mahdollista. Ennen kuin yksityisiä tietoja lähetetään, tulisi vastaanottaja todentaa.
- Jos on pakko käyttää implisiittisiä Intentejä, tulisi niitä suojellakseen määritellä vahvat oikeudet. Muiden komponenttien palauttamat tulokset tulisi todentaa, että voidaan olla varmoja niiden tulleen oikeasta lähteestä. Toiminnat tulisi tehdä vasta tämän jälkeen.
- Komponenttien tulisi olla yksityisiä, paitsi jos niitä ei ole erityisesti suunniteltu käsittelemään muiden komponenttien pyyntöjä. Kehittäjien tulisi olla tietoisia, että määrittelemällä komponentille Intent-suodattimen, tekee se siitä samalla julkisen ja haavoittuvamman hyökkäyksille. Suodattimet eivät ole turvallisuusmekanismi ja ne voidaan ohittaa käyttämällä eksplisiittisiä Intentejä.
- Kriittisiä ja sovelluksen tilaa muuttavia toimintoja ei tulisi toteuttaa julkisissa komponenteissa. Jos komponentin täytyy käsitellä sekä yksityisiä että julkisia pyyntöjä, tulisi se mahdollisesti jakaa kahteen osaan.
- Signature- ja Signature OrSystem-oikeuksia vaatimalla voidaan komponentin altistumista rajata luotettuihin sovelluksiin, koska muilla sovelluksilla ei voi näitä oikeuksia olla.

Mietittäessä puolustautumiskeinoja on syytä muistaa myös tietoturvan perusperiaatteet, kuten Six [2011] tuo kirjassaan esille: Sovelluksen tulisi tallentaa ainoastaan tarvittavat tiedot ja yksityiset tiedot tulisi olla salatussa muodossa, jolloin väärinkäyttäjä ei pääse hyödyntämään niiden sisältöä. Tämä useita suojauskerroksia sisältävä tietoturvamalli tunnetaan myös nimellä Defense in Depth eli DiD. Sama pätee itse luotuihin oikeuksiin: Kunkin oikeuden tulisi vapauttaa mahdollisimman vähän toimintoja haltijalleen, eli esimerkiksi luku- ja kirjoitusoikeudet tulisi olla erilliset.

5. Yhteenveto

Android-käyttöjärjestelmän suosio on kasvamassa; jo nyt se on maailman yleisin käyttöjärjestelmä älypuhelimissa [Gartner, 2011]. Android Market sisältää valtavan määrän, yli 500 000, sovellusta, joita käyttäjät voivat ladata mobiililaitteisiinsa [research2guidance, 2011]. Koska periaatteessa kuka tahansa voi julkaista itse tekemiään sovelluksia siellä, on joukossa myös haavoittuvia ohjelmia ja pelejä, joita haittasovellukset pääsevät hyväksi käyttämään. Käsittelin tässä tutkielmassa yhtä osa-aluetta Android-sovelluksissa, jota haitalliset sovellukset voivat hyväksi käyttää. Chin ja muut [2011] tutkivat kaksikymmentä suosittua Android-sovellusta, jotka koostuivat niin maksullisista kuin ilmaisistakin ohjelmista. Heidän tutkimuksensa osoitti, että näistä kahdestakymmenestä suositusta sovelluksesta 45 % sisälsi Intenteihin liittyviä haavoittuvuuksia. Tämä antaa viitteitä siitä, että tietoturvaongelma on laajalle levinnyt, ja että sovellusten kehittäjien tulisi kiinnittää nykyistä enemmän huomiota siihen, miten heidän ohjelmansa komponentit kommunikoivat keskenään ja muiden sovellusten kanssa, koska heidän luomansa ohjelmat voivat mahdollistaa haittaohjelmien toiminnan. Intenteihin liittyvät tietoturvaongelmat ovat kuitenkin ratkaistavissa sovellusten paremmalla suunnittelulla ja toteutuksella, kuten tässä tutkielmassa on tuotu esille.

Haittaohjelmille on olemassa myös muita keinoja hankkia Android-sovellusten käyttöoikeudet ilman laitteenkäyttäjän suostumusta, joita en käsitellyt tässä tutkielmassa. Esimerkiksi Davi ja muut [2011] pystyivät lähettämään tekstiviestejä sovelluksella, jolle ei siihen ole myönnetty käyttäjän toimesta oikeuksia. He käyttivät hyödykseen laillisen, mutta haavoittuvan, sovelluksen muistin ylivuotovirhettä. Mm. tällaisten Android-käyttöjärjestelmän tietoturva-aukkojen hyväksikäyttö ja korjaaminen voisi olla tutkielmani aihe tulevaisuudessa.

Viiteluettelo

[Chin *et al.*, 2011] Erika Chin, Adrienne Porter Felt, Kate Greenwood and David Wagner, Analyzing inter-application communication in Android. In: *Proc. of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, (2011). Also available as http://www.cs.berkeley.edu/~afelt/intentsecurity-mobisys.pdf.

[Davi *et al.*, 2011] Lucas Davi, Alexandra Dmitrienko, Ahmad-Reza Sadeghi and Marcel Winandy, Privilidge escalation attacks on Android. In: *Proc. of the 13th International Conference on Information security (ISC 2010), Lecture Notes in Computer Science* **6531** (2011), 346-360.

[Enck *et al.*, 2009] William Enck, Machigar Ongtang and Patrick McDaniel, Understanding Android security, *IEEE Security & Privacy* **7** (2009), 10-17.

[Gartner, 2011] Gartner, Gartner Says Sales of Mobile Devices Grew 5.6 Percent in Third Quarter of 2011; Smartphone Sales Increased 42 Percent. http://www.gartner.com/ it/page.jsp?id=1848514. Checked 12.19.2011.

[Meier, 2010] Reto Meier, *Professional Android 2 Application Development*. John Wiley and Sons, 2010.

[Ongtang *et al.*, 2009] Machigar Ongtang, Stephen McLaughlin, William Enck and Patrick McDaniel, Semantically Rich Application-Centric Security in Android. In: *Proc. of the Annual Computer Security Applications Conference* (ACSAC 2009), 340-349.

[Porter Felt *et al.*, 2010] A. Porter Felt, Kate Greenwood and David Wagner, The Effectiveness of Install-Time Permission Systems for Third-Party Applications, University of California, Electrical Engineering and Computer Sciences, Report **UCB/EECS-2010-143,** December 3, 2010. Also available as http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-143.pdf.

[research2guidance, 2011] research2guidance, Android Market reaches half a million successful submissions. http://www.research2guidance.com/android-market-reaches-half-a-million-successful-submissions. Checked 12.20.2011.

[Six, 2011] Jeff Six, *Application Security for the Android Platform*. O'Reilly, 2011.

Koodiesimerkit

```
Intent intent = new Intent(this, AlarmClockBroadcastReceiver.class);
intent.setData(selectedAudio);
intent.setAction(ACTION_START_ALARM);

mAlarmIntent  =  PendingIntent.getBroadcast(this, 0, intent, PendingIntent.FLAG_CANCEL_CURRENT);
mAlarmManager.set(AlarmManager.RTC_WAKEUP, System.currentTimeMillis() + timeFromNow, mAlarmInte
```

Koodi 1: Herätyksen asettaminen. AlarmClockBroadcastReceiver saa AlarmServicen lähettämän broadcastin asetettuna aikana.

```
public class AlarmClockBroadcastReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context pContext, Intent pIntent) {

        if(pIntent.getAction().equals(AlarmClockSettingsActivity.ACTION_START_ALARM)){
            pIntent.setClass(pContext, AlarmClockActivity.class);
            pIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            pContext.startActivity(pIntent);
        }

    }
}
```

Koodi 2: BroadcastReceiver, joka vastaanottaa AlarmServicen lähettämän broadcastin. Jos broadcast sisältää ACTION_START_ALARM-actionin, käynnistetään herätyksen aikanen AlarmClockActivity.

Ohjelmistoprojektin mittaamisesta

Heikki Santasalo

Tiivistelmä.
Ohjelmistotyötä voidaan mitata esimerkiksi tuottavuuden ja laadun perusteella. Laadun mittarit voidaan jaotella ohjelmiston oikeellisuutta, ylläpidettävyyttä, yhtenäisyyttä ja käytettävyyttä mittaaviin mittareihin. Erityyppisiä mittareita voidaan arvioida muun muassa niiden relevanttiuden, tosiaikaisuuden, objektiivisuuden ja luotettavuuden suhteen. Mittaamisen merkitys on korostunut viime vuosina yleistyneissä adaptiivisuuteen nojaavissa ketterissä menetelmissä, joissa toimintaa pyritään jatkuvasti kehittämään tehtyjen havaintojen perusteella. Samaan aikaan kuitenkin ohjelmistoprojektin mittaamista pidetään hyvin arkana ja vaikeana aiheena.

Avainsanat ja -sanonnat: Ohjelmistokehitys, Scrum, mittarit, mittaaminen.
CR-luokat: D.2.8

1. Johdanto

Ketteristä menetelmistä on muutamassa vuodessa tullut yleisin tapa tehdä ohjelmistotyötä. Niiden käyttöä perustellaan vetoamalla tuottavuuden kasvuun ja korkeampaan laatuun sekä nopeampaan ohjelmiston toimitusaikaan. Ohjelmistotyö kuitenkin on yksi alueista, jonka täsmällistä mittaamista on perinteisesti pidetty mahdottomana tai ainakin hyvin haasteellisena. Niinpä herää kysymys, miten mitata parantunut tehokkuus sekä korkeampi laatu ja näin varmistaa ketterillä menetelmillä aikaansaadut parannukset? Tämän työn tarkoituksena on tutkia, millaisia ohjelmistotyön mittaamistapoja on olemassa sekä arvioida niiden kelvollisuutta ketteriä menetelmiä, erityisesti Scrumia, käyttävässä projektissa. Työn ulkopuolelle rajataan tilastollisten menetelmien käyttö.

Adolph [2006] määrittelee ketterän projektin tai organisaation sellaiseksi, joka pystyy havaitsemaan muutoksen ja reagoimaan siihen nopeammin kuin toimintaympäristö muuttuu. Esimerkkinä ketterästä projektista Adolph [2006] antaa hieman tavallisuudesta poikkeavan tapauksen – toisessa maailmansodassa esitellyn salamasodan. 1930-luvulla ranskalaiset rakensivat Maginot-puolustuslinjan Saksaa vastaan. Ensimmäisen maailmansodan kokemuksiin perustuen puolustuslinja oli voittamaton. Toiseen maailmansotaan tulessa sodankäynti oli kuitenkin muuttunut ns. salamasodaksi, ja nopeasti liikkuvat saksalaiset pystyivät kiertämään koko puolustuslinjan. Valtavaa Maginot-linjaa ei voinut siirtää paikkaan, jossa se olisi ollut hyödyllinen muuttuneessa tilantees-

sa, eli Ranska ei kyennyt reagoimaan toimintaympäristön muutokseen, ja oli pian tuhon oma.

Kuten jo edellinen esimerkki osoittaa, ketterästi on toimittu kauan, vaikkakaan toimintatapoja ei ole kutsuttu "ketteriksi". Varsinaiset ketterät menetelmät ohjelmistotuotannossa ovat kuitenkin vielä verrattain uusi keksintö – niiden käyttö on yleistynyt vasta aivan viime vuosina. Ns. agile manifesto, jossa 17 ketterän kehityksen puolestapuhujaa julisti ketterän ohjelmistokehityksen arvot, annettiin vasta vuonna 2001. Manifestin antamisen jälkeen ketterien menetelmien leviäminen on ollut nopeaa – on arvioitu että vuonna 2010 90% ohjelmistoalan yrityksistä käytti ketteriä menetelmiä jossakin määrin, ja että yli puolet ohjelmistoprojekteista oli toteutettu ketteriä menetelmiä käyttäen [Agile, 2010].

Ketterät menetelmät on kehitetty vastaukseksi perinteisen ohjelmistokehitysmenetelmän eli ns. vesiputousmallin ongelmille. Suurimpina ohjelmistokehityksen ongelmina on nähty myöhästyneet tuotteet, joiden laatu on huono [Pressman, 1997; Haikala ja Märijärvi, 2000]. Ongelmat ovat olleet niin suuria, että on puhuttu jopa ”ohjelmistokriisistä” [Haikala ja Märijärvi, 2000]. Ketteriä menetelmiä kokeillaan, koska niiden uskotaan nostavan tuottavuutta, parantavan laatua ja parantavan kykyä sopeutua muutoksiin [Agile, 2010]. Scrumin kehittäjän Ken Schwaberin [2002] mukaan Scrum tekee tiimeistä ”hyperproduktiivisia” ja saa niiden tuottavuuden kasvamaan eksponentiaalisesti. Cohnin [2009] mukaan ketterät menetelmät nopeuttavat tuotteiden saamista markkinoille, nostavat laatua sekä tuottavuutta ja parantavat niin työntekijä- kuin asiakastyytyväisyyttäkin. Ketteriä menetelmiä kokeilleista ja State of Agile Surveyhin [2010] vastanneista tuottavuutta pystyi parantamaan 74%, laatua nostamaan 65% ja muutoksiin sopeutumaan paremmin 87%. Cohn [2009] on saanut samankaltaisia tuloksia – suurin osa ketteriä menetelmiä kokeilleista on saanut niistä hyötyä.

2. Ketterien menetelmien käytännöt

Ketteryys on ennen kaikkea toimintaperiaate, mutta varsinaiset ketteriä menetelmiä käyttävät projektit ovat tunnistettavissa tiettyjen työtapojen perusteella. State of Agile Surveyn [2010] lajitteluperusteina ovat työtavat kuten iteratiivinen ohjelmistoprosessi ja iteraation suunnittelupalaveri, päivittäiset tilannepalaverit, yksikkötestaus ja jatkuva integrointi, pariohjelmointi, retrospektiivi, avoin työtila ja säännöllinen koodin refaktorointi.

Eri projekteissa käytetään vaihtelevaa määrää näistä. "Ketterät menetelmät" ei tarkoitakaan ainoastaan yhtä tapaa tehdä ohjelmistokehitystä. Alalajikkeita on kymmeniä; tärkeimpinä mainittakoon Scrum, XP, RUP, Lean, TDD, Kanban.

Rajanveto eri metodien välillä on vaikeaa. Koska Scrum on ennemminkin toimintatapakehys kuin tiukka menetelmä, voi projekti käyttää esimerkiksi yhtä aikaa XP:n menetelmiä Scrumin sisällä. State of Agile Surveyn [2010] mukaan 58% ketteriä menetelmiä käyttävistä projekteista käyttää Scrumia. Jos mukaan lasketaan projektit, jotka käyttävät yhtäaikaisesti sekä Scrumia että XP:tä, osuus on jo 75%. Scrumin yleisyydestä johtuen tässä tutkielmassa keskitytään lähinnä siihen.

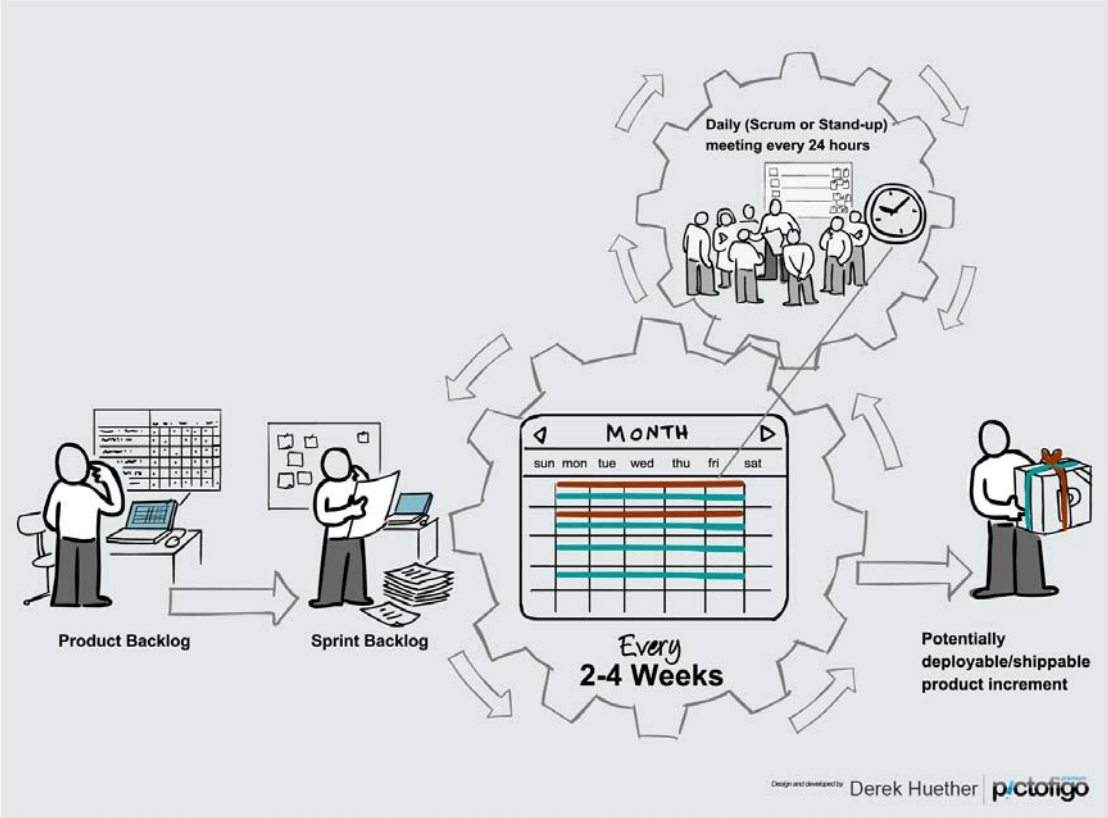
Scrum-menetelmän nimi tarkoittaa rugby-pelin vaihetta, jossa peli avataan uudelleen pallon jouduttua pois pelistä [Schwaber and Beedle, 2002]. Jääkiekossa termi on vakiintunut kuvaamaan laidassa tai kulmassa tapahtuvaa pelitilannetta, jossa kiekko on jumissa ja kumpikin joukkue yrittää saada sen itselleen. Tilanteen ymmärtäminen avaa myös Scrum-prosessia hyvin – oikeastaan kaikki sen käytännöt pyrkivät poistamaan ulkopuolisia esteitä tuottavan työn tekemiseltä sekä varmistamaan, että työ ei keskeydy myöskään minkäänlaisten työhön liittyvien ongelmien takia.

Scrum-menetelmässä on kolme erilaista roolia. Scrum-tuotekehityksen ytimen muodostaa tiimi, johon kuuluu ohjelmistosuunnittelijat sekä testaajat. Tiimiä valmentaa, sen toimintaa fasilitoi ja sille tulevia esteitä poistaa ScrumMaster. ScrumMaster voi olla myös osa tiimiä [Scrum Guide]. Tuoteomistajan (en. *product owner*) tehtävä on priorisoida tehtävät työt siten, että projekti on mahdollisimman arvokas asiakkaalle. Tuoteomistaja myös edustaa kaikkia projektia ohjaavia sidosryhmiä [Schwaber, 2004].

Tekemättä oleva työ on tuotteen työlistalla (en. *product backlog*), joka on priorisoitu lista töitä. Työlistalla olevat asiat voivat olla periaatteessa mitä tahansa, mutta tyypillisesti ne ovat ns. käyttäjätarinoita (en. *user story*), lyhyitä käyttötapauskertomuksia. Toteutettavien ominaisuuksien kuvaaminen käyttäjäkertomuksina formaalien vaatimusten sijaan varmistaa, että kaikki tehtävä työ todella lisää käyttäjän saamaa arvoa. Tuotteen työlistasta vastaa tuoteomistaja; hän voi muuttaa sitä milloin tahtoo [Schwaber, 2004].

Tuotteen työlistaa muutetaan tuotteeksi kahdessa sisäkkäisessä silmukassa. Iteraatiot ovat sprinttejä, 1-4 viikon mittaisia pyrähdyksiä, joihin sprintin alussa lohkaistaan palanen tuotteen työlistan päältä. Näiden iteraatioiden sisällä työ jäsentyy päivittäisten Daily Scrumien ympärille, joissa tiimi kokoontuu synkronoimaan tekemisensä. Molemmat silmukat ovat ns. palautesilmukoita – sprintin päätteessä tiimi esittelee työnsä tulokset sidosryhmille, projektin tila arvioidaan ja jatketaan uuteen sprinttiin. Daily Scrumissa tiimi kokoontuu päivittäin tutkimaan omia ja toisten tiimiläisten meneillään olevia töitä ja mukauttaa toimintansa havaintojensa mukaisesti. Näitä silmukoita havainnollistaa ku-

va 1, jossa alempi ratas on kahdesta neljään viikkoon kestävä sprintti ja ylempi ratas on päivittäin toistuva Daily Scrum -silmukka.



Kuva 1. Scrum-prosessi kaaviona-

Sprintin suunnittelutapaaminen (en. *sprint planning*) on joka iteraation alussa pidettävä kokous, jossa tiimi ottaa tuotteen työlistan päältä sopivaksi katsomansa määrän asioita, jotka se sitoutuu tekemään valmiiksi seuraavan sprintin aikana [Schwaber, 2004]. Näin syntyy sprintin työlista. Sprintin työlistaa saa muokata vain tiimi, ja sekin vain suunnittelutapaamisen ajan. Sen jälkeen sprintin työlistaa saa muuttaa vain taskien eli sprintin työlistalle valittujen asioiden osittamisen osalta [Scrum Guide].

Sprintin alettua tiimi jätetään rauhaan toteuttamaan valitsemansa työlistan työt parhaaksi näkemällään tavalla [Schwaber, 2004]. Projektipäällikkö ei siis jaa tehtäviä, eikä ScrumMaster kerro, mitä seuraavaksi pitäisi tehdä – ainoa mikä ohjaa tiimiä, on sen yhteinen sitoumus tehdä valmiiksi sprintin työlistalla olevat työt. Päivittäin tiimi kokoontuu synkronoimaan ja suunnittelemaan tekemisensä Daily Scrumiin, jossa jokainen tiimiläinen vastaa kolmeen kysymykseen [Schwaber, 2004]:

1) mitä olet tehnyt edellisen Daily Scrumin jälkeen?

2) mitä aiot tehdä seuraavaan Daily Scrumiin mennessä?

3) estääkö joku asia työn suorittamista?

Viimeiseen kysymykseen vastatut asiat ovat esteitä, joiden poistaminen kuuluu keskeisesti ScrumMasterin tehtäviin. Scrum-ideologian mukaisesti este on poistettava nopeasti käyttäen mitä tahansa tarvittavia keinoja [Scrum Guide] – estynyt tuottava työ on kuin pysähtynyt rugby-peli tai laitaan jumiin jäänyt jääkiekko, tilanne josta ei pystytä tekemään maalia eli tulosta.

Sprintin lopputulos on potentiaalisesti julkaisukelpoinen tuote – tuotteen on siis aina oltava siinä kunnossa, että se voitaisiin julkaista, mutta aina niin ei tehdä. Tämä tarkoittaa, että pelkkä koodin kirjoittaminen ei riitä – ohjelmisto pitää myös dokumentoida, testata, integroida ja paketoida [Schwaber, 2004].

Testiautomaatio on yksi ketterän ohjelmistokehityksen kulmakivistä. Aiemmin jo todettiin, että State of Agile Survey [2010] tunnistaa ketterät projektit mm. yksikkötestauksen ja jatkuvan integroinnin perusteella. Perinteisessä vesiputousmallissa testaaminen tapahtuu projektin loppupuolella, kun tuote on jo lähes valmis. Tästä syystä mahdolliset laadulliset heikkoudet löydetään tyypillisesti vasta vaiheessa, jossa niille on myöhäistä tehdä enää mitään [Lewis, 1998]. Jatkuva integrointi taas tarkoittaa, että aina kun ohjelmakoodia lisätään versionhallintajärjestelmään, se myös integroidaan ja testataan, alkaen yksikkötestauksesta ja päättyen järjestelmätestaukseen. Yksikkötestaus on ohjelmiston pienintä mahdollista yksikköä koskeva testi [Haikala ja Märijärvi, 2000], olio-ohjelmoinnissa testattavana on tyypillisesti yksi luokka. Kun joka yksikölle on laadittu kattavat yksikkötestit, voidaan niitä muuttaa turvallisesti, automaattisten testien paljastaessa, jos muutosten sivutuotteena jokin toiminnallisuus hajosi. Automatisoitu testaus ja jatkuva integrointi toimii siis myös eräänlaisena turvaverkkona tai ohjelmistotuotteen liikennevaloina.

Sprintin päätteeksi sprintti katselmoidaan. Tiimi esittelee valmiiksi saadun toiminnallisuuden projektin sidosryhmille. Tämän perusteella sprintti joko hylätään tai hyväksytään. Projektin sidosryhmät myös arvioivat, mikä on projektin tila ja vaihe, mitkä asiat ovat muuttuneet ja miten ne vaikuttavat projektiin. Tuotteen työlista on priorisoituna, ja niin seuraava sprintin suunnittelutapaaminen voi alkaa [Scrum Guide].

Tyypillisesti sprintin loppuvaiheessa on myös sprintin retrospektiivi. Siinä tiimi pohtii mennyttä sprinttiä, sen onnistumisia ja vaikeuksia, sekä kehittelee parannusehdotuksia. Parannusehdotusten tulisi koskea ensisijaisesti tiimiä, ihmisiä ja näiden vuorovaikutusta sekä Scrum-prosessin ja ketteryyden toteutusta. Nämä ehdotukset pyritään toteuttamaan seuraavan sprintin aikana, jonka jälkeen arvioidaan niiden vaikutusta. [Schwaber, 2004; Scrum Guide].

Sprintti tuottaa monenlaista dataa tehdyistä asioista. Jo ennen sprintin suunnittelua tiimi on arvioinut kaikkien työlistalla olevien töiden työmäärän

karkealla tasolla. Sprintin suunnittelutapaamisessa tiimi jakaa kunkin sprintin työlistan työn taskeihin, pienempiin työtehtäviin, ja antaa näille työmääräarviot. Taskien työmäärää päivitetään jatkuvasti kunnes taski on saatu valmiiksi. Näistä tuntiarvioista ja niiden muutoksesta saadaan burndown-kaavio, joka näyttää, kuinka paljon sprintissä työtä on tekemättä suhteessa jäljellä olevaan aikaan. Työlistan töiden karkeasta arviosta saadaan sprintin ja tiimin nopeus (en. *velocity*), joka kertoo, kuinka suureksi arvioimansa työmäärän tiimi on saanut sprintin aikana valmiiksi. Laskemalla eri sprinteissä toteutuneita nopeuksia voidaan arvioida sitä, missä ajassa tuotteen työlistalla olevia asioita saadaan tehdyksi.

3. Ketterien menetelmien periaatteet

Ketterien menetelmien tärkeimmät periaatteet ovat empiirinen prosessikontrolli, itseohjautuvuus, ryhmätyöskentely ja aikasidonnaisuus (en. *timeboxing*). Ns. Agile manifestossa [Beck et al., 2001] ketterä yhteisö määritteli ketterän ohjelmistokehityksen arvot, joiden mukaan ketterä ohjelmistokehitys painottaa seuraavia arvoja:

- yksilöitä ja vuorovaikutusta enemmän kuin prosesseja ja työkaluja
- toimivaa sovellusta enemmän kuin kokonaisvaltaista dokumentaatiota
- asiakasyhteistyötä enemmän kuin sopimusneuvotteluita
- muutokseen reagoimista enemmän kuin suunnitelman noudattamista.

Agile manifeston [Beck et al., 2001] mukaan ohjelmistokehityksessä oleellisinta on siis toimiva ohjelmisto ja erityisesti sen asiakkaalle tuottama lisäarvo. Näihin päästään tekemällä ohjelmistokehitystä yksilöä kunnioittaen ja tiimityötä tukien tiiviissä yhteistyössä asiakkaan kanssa. Manifesti näkee muutokseen reagoimisen arvokkaampana kuin suunnitelman noudattamisen – muutos on mahdollisuus ja realiteetti, ei katastrofi.

3.1. Empiirinen prosessikontrolli

Empiirinen prosessikontrolli on määritellyn prosessikontrollin (en. *defined process control*) vastakohta. Määritelty prosessi sopii yksinkertaiseen työhön, joka toistuu aina pääosin samanlaisena, esimerkiksi auton kokoonpano tai rakennusprojekti. Ohjelmistokehitystyötä ei tälläisena voi pitää – se on ennemminkin luovaa tutkimustyötä [Schwaber and Beedle, 2002]. Koska ohjelmistoprojektin kompleksisuus on tyypillisesti hallitsematon [Schwaber, 2004] ja koska jokainen projekti on erilainen [Schwaber and Beedle, 2002], tarvitaan aina uudentyyppistä osaamista. Tästä syystä toimintaa määräävästä prosessista on vain vähän hyötyä. Sitä vastoin ohjelmistoprojektin etenemisen tarkkanäköisen havainnoinnin ja toiminnan sopeuttamisen osaamisesta taas on paljonkin hyötyä.

Empiirinen prosessikontrolli tarkoittaa adaptiivisuutta, eräänlaista palautesilmukkaa, jossa toisiaan jatkuvasti seuraavat havainnointi - sopeutuminen - päätös - toiminta (Observation-Orientation-Decision-Action (OODA)). Tehdyn päätöksen jälkeen toteutetun toiminnan tulokset havainnoidaan. Sopeutumisella ymmärretään tässä prosessia, jossa havainto analysoidaan. Analyysiin vaikuttaa varsinaisen havainnon lisäksi kulttuurinen ja geneettinen perintö, olosuhteet sekä aiempi henkilökohtainen kokemus [Adolph, 2006]. Adolph [2006] esittää havaintonsa sotilaallisesta näkökulmasta – saksalaiset suorittivat palautesilmukkansa aina ranskalaisia nopeammin, ja näin onnistuivat luomaan epäjärjestystä ja pakokauhua ranskalaisiin – mutta yritysmaailmassa kyse on samasta ilmiöstä, viholliset vain ovat kilpailijoita, aikatauluja tai yksinkertaisesti yleinen halu kehittyä.

Ketterässä projektissa tai organisaatiossa empiriaa sovelletaan yksilöön, tiimiin, tiimityöhön, ohjelmistoon, ohjelmistotyöhön, organisaatioon jne. Yleisimmin tämä esiintyy ketterän tiimin päivittäisissä statuspalavereissa, joissa tiimi synkronoi toimintansa ja koordinoi seuraavan päivän tekemisensä. Kokonainen iteraatio edustaa taas laajempaa palautesilmukkaa; iteraation päätteeksi tehty tuote esitellään asiakkaalle, joka päättää jälleen kerran, mikä on seuraavaksi tärkeintä. Empiirinen prosessikontrolli tarkoittaa joustavuutta, mukautuvuutta ja tuottavuutta [Schwaber and Beedle, 2002]. Tässä mielessä ketteryyden astetta organisaatiossa mittaa palautesilmukan nopeus sekä tiheys.

3.2. Itseohjautuvuus ja ryhmätyö

Toinen tärkeä ketterien menetelmien periaate on itseohjautuvuus (en. *self-organization*). Agile manifeston [Beck et al., 2001] mukaan parhaat arkkitehtuurit, suunnitelmat ja vaatimukset nousevat esiin itseohjautuvista tiimeistä. Itseohjautuvuus tarkoittaa tiimin kykyä ratkaista ongelmat ja järjestää toimintansa parhaalla mahdollisella tavalla itse, kun toiminnan tavoite on selvillä. Väinö Linnan Tuntematon sotilas -teoksesta löytyy hyvin itseohjautuvuutta kuvaava kohtaus, kun vänrikki Koskelan tehtävänä on saada miehensä pienelle kuormaauton lavalle:

"- miten helvetin tavalla tuonne sopii joukkue? Joku jäi ihmettelemään, mutta huomasi sitten, että myöhästyminen tietää huonoa sijoitusta ja ryntäsi mukaan. Koskela katseli äänettömänä kuormausta peukalot vyön alle työnnettyinä. Hän tiesi, että elävä kuorma järjestyisi itsestään parhaalla mahdollisella tavalla, eikä sen vuoksi puuttunut koko asiaan."

Voi helposti kuvitella, miten vaikeaa olisi käskyttää kolmellekymmenelle miehelle jokaiselle oma paikkansa sekä tapa, jolla jokainen saa varusteensa aseteltua optimaalisesti. Samoin ohjelmistoprojektissakin voi esimerkiksi ohjeistaa tiiminsä tekemään ponnahdusikkunan x, joka aukeaa tekstikentästä y, ja johon

haetaan modulista z kuvatunlaiset ehdot täyttävät kontaktit, jotka piirretään erikseen määriteltyjen ohjeiden mukaan. Tai sitten voi kertoa, että ohjelmiston käyttäjä haluaa vastaanottajien nimiä kirjoittaessaan pystyä valitsemaan kirjoitettuun tekstiin täsmäävät yhteystiedot. Jälkimmäistä vaihtoehtoa puoltaa kolme seikkaa. Ensinnäkin, koska ketterissä menetelmissä kaikkea tekemistä arvioidaan käyttäjälle lisätyn arvon perusteella [Beck et al., 2001], on järkevämpää tehdä niin kuin asiakas tahtoo, eikä niin kuin projektipäällikkö tahtoo. Toiseksi, asiantuntijat eivät reagoi työtehtävien pikkutarkkaan ohjaamiseen hyvin [Augustine et al., 2005]. Ja kolmanneksi, aiemmin mainitun kompleksisuuden takia projektipäällikön kyky eritellä ja kertoa jokainen työvaihe eksplisiittisesti on todennäköisesti hyvin rajallinen verrattuna ohjelmistotiimin kollektiiviseen kykyyn tuottaa ratkaisu itsenäisesti [Augustine et al., 2005; Schwaber and Beedle, 2002].

Itseohjautuvuus tarkoittaa yleensä myös eräänlaisen organisatorisen rakenteen syntymistä. Tiimi päättää itse sisäisen vastuunjakonsa, asiantuntijansa, tapansa suunnitella, toteuttaa, testata jne. Jokin tiimi esimerkiksi voi pitää kaiken vastuun koko tiimillä, ja joku taas voi sisältää erilaisia asiantuntijoita. Jokin tiimi voi tehdä työnsä ottaen monta tehtävää kerrallaan kun toinen tiimi taas tekee yhden tehtävän kerrallaan koko tiimin voimin [Cohn, 2009].

Itseohjautumisen kannalta oleellista on, että tiimillä on yhteinen maali (saada joukkue lavalle) ja yhteiset pelisäännöt tai rajoitteet (mukaan otettavat tavarat, lavan koko, jne). Tärkeää on myös, että tiimin kesken vallitsee luottamus ja kunnioitus, niin että maaliin pyritään yhdessä eikä kukin yksinään. Luottamusta tarvitaan myös tiimin ulkopuolelta – on tärkeää, että tiimin ulkopuoliset henkilöt ja joissakin tapauksissa jopa ScrumMaster luottavat tiimin kykyyn hoitaa tehtävänsä, ja antavat tiimin ratkaista ongelmansa ja tehdä päätöksensä itse [Schwaber, 2004; Adolph, 2006].

3.3. Aikasidonnaisuus

Scrumin aikasidonnaisuus tarkoittaa, että kaikkien toimintojen kesto, aina päivittäisestä Scrum-kokouksesta lähtien, on tarkoin määritelty. Käytännössä tämä ilmenee siten, että toiminnasta voidaan tinkiä, mutta ei ajasta. Jos esimerkiksi Daily Scrum uhkaa kestää yli viisitoista minuuttia, se katkaistaan kesken ja yritetään tulla toimeen vajavaisilla tiedoilla. Tai jos sprintin sisältö ei ole sprintin päättyessä valmis, sprintti päätetään silti ja seuraavassa sprintin suunnittelutapaamisessa harkitaan, miten on parasta jatkaa. Tämä pakottaa keskittymään oleelliseen ja käsittelemään tärkeät asiat ensimmäiseksi – mikään ei kohdista mieltä niin kuin hirttosilmukka [Schwaber, 2004].

4. Miksi ohjelmistotyötä mitataan?

Puhuttaessa ohjelmistotyön mittaamisesta ja sen syistä saattaa ensimmäisenä tulla mieleen synkkiä kuvia suorituskeskeisestä ja ihmisiä mustavalkoisesti jaottelevasta yrityskulttuurista. Ohjelmistotyöläiset näkevät ammattinsa eräänlaisena käsityönä, jonka kulttuurissa ulkopuolinen valvonta ja ohjaus, puhumattakaan työtehtävien yksityiskohtaisesta ohjaamisesta, aiheuttaa voimakkaita vastareaktiota [Haikala ja Märijärvi, 2000; Augustine et al., 2005]. Vastareaktiot johtunevat viha-rakkaus -suhteesta mittareihin, ja tyypillisin reaktio on mittaamisen mahdottomuuden tai ongelmien esiintuominen. Jo ohjelmistotuotteen, prosessin tai laadun mittaamisesta puhuminen onkin hankalaa, mutta itse ohjelmistotyön mittaaminen on vielä vaikeampi aihe [Berard]. Vaikeudestaan sekä väärinkäyttömahdollisuuksistaan huolimatta mittaaminen on kuitenkin monista syistä välttämätöntä [Haikala ja Märijärvi, 2000].

Projektin seuranta on yksi mittareiden käyttökohde. On tärkeää tietää paitsi projektin etenemisvauhti myös mahdolliset ongelmakohdat, ja monesti perinteiset, projektin tai organisaation laajuiset seurantamenetelmät ovat melko karkean tason mittareita, jotka eivät pureudu ongelmatasolle asti [Pfleeger, 1993]. Vaikka koko projekti edistyisi tavoitteiden mukaisesti, jokin yksittäinen tiimi voi silti olla suurissa vaikeuksissa, jotka voivat viiveellä heijastella muiden tekemisiin.

Yleensä halutaan seurata myös ohjelmistokehitysprosessien tilaa ja tehokkuutta [Pressman, 1997]. Esimerkiksi yritys, joka on valinnut ketterät menetelmät tavakseen tehdä ohjelmistotyötä, luonnollisesti haluaa selvittää ja seurata, miten ketterästi projektit tekevät töitä [Cohn, 2009]. Tai laajemminkin: jos yritys haluaa parantaa toimintaansa erilaisilla kehittämistoimenpiteillä ja hankkeilla, miten se voi arvioida toimenpiteiden vaikutusta ja tarvetta ilman jonkinlaisia mittareita [Haikala ja Märijärvi, 2000]?

Ohjelmiston laaduntarkkailu ja laadun parantaminen on yksi keskeinen mittaamisen kohde [Pressman, 1997]. Ohjelmiston laatuun liittyviä mittareita useimmissa ohjelmistoprojekteissa onkin. Jotkut mittarit voivat olla myös sellaisia, jotka paitsi kertovat nykytilasta, myös enteilevät tulevaa. Pfleeger [1993] esittelee artikkelissaan luokkitelupuuanalyysin, jossa projektia käydään läpi kysymyssekvenssien kautta. Tällaisella mittarilla voidaan esimerkiksi löytää moduuleita, joiden odotetaan olevan alttiimpia virheille. Esimerkkinä tällaisesta voisi olla luokka, jossa on enemmän kuin 300 riviä koodia, jolle ei ole tehty katselmointia ja jota on muutettu enemmän kuin viisi kertaa.

Palkitsemisjärjestelmät ovat yksi mittaamisen motivaatio. Yritys, joka haluaa palkita työntekijöitä näiden työtehtävissään suoriutumisen mukaan, on kiin-

nostunut mittaamaan ohjelmistoprojektin, -tiimin tai -työntekijän suoritustasoa sekä muutosta näissä.

5. Mittareiden arvioinnista

Lewis [1998] määrittelee hyvän mittarin tunnusmerkeiksi relevanttiuden, kokonaisuuden (en. *completeness*), tosiaikaisuuden sekä taloudellisuuden. Haikala ja Märijärvi [2000] lisäävät näihin selektiivisyyden, objektiivisuuden, luotettavuuden sekä taloudellisuuden. Cohn [2009] taas painottaa vahvasti lähinnä yksinkertaisia mittareita, koska pitkän ajan kuluessa hienommiksi viriteltyjen mittareiden hyödyt yksinkertaisiin verrattuna katoavat.

Relevanttius tarkoittaa, että mittari on oikein asetettu. Mitattavien asioiden valikoiminen on suurelta osin arvovalinta. Mittareiden asettajalla pitää ensin olla selkeä näkemys siitä, minkälainen ohjelmistokehitys on hyvää. Ohjelmistotyötä voidaan arvottaa esimerkiksi tuottavuuden, laadun tai saadun lisäarvon suhteen. Laatu on tässä yhteydessä ymmärrettävä ohjelmiston korkeana laatuna, joka sekin on määrittelykysymys – Haikalan ja Märijärven [2000] esittämä määritelmä laadulle tarkoittaa vain tuotteen ja toiminnan mitattavia ominaisuuksia. Tuntematta toimintaympäristöä ja mittaamisen tavoitteita onkin siis mahdotonta sanoa, onko mittari relevantti.

Kokonaisuudella tarkoitetaan, että mittaristo kattaa kaikki projektin kannalta tärkeät seikat. Kun mitattavat henkilöt pyrkivät optimoimaan mitattavat asiat niiden asioiden kustannuksella, jotka mittareihin eivät vaikuta [Lewis, 1998], niin mittarit pitää valita niin, että ne mittaavat hyvää ohjelmistokehitystä, eivätkä pelkästään jotakin välillistä tunnuslukua. Esimerkkinä tyypillisestä välillisestä tunnusluvusta on tuottavuuden mittaamisessa yleisesti käytetty tuotettujen koodirivien määrä. Mitattaessa ainoastaan tuotettujen koodirivien määrää on todennäköistä, että projektin työntekijät saavat paljon koodirivejä aikaan, mutta ohjelmiston laatu on huono. Koska yksittäinen mitattava asia harvoin riittää kokonaisuusvaatimuksen täyttämiseen, on hyvä mittari yleensä siis jonkinlainen yhdistelmä erilaisista tunnusluvuista [Berard].

Mittarin tosiaikaisuus tarkoittaa sitä, että mittarin tuottama data on käytettävissä niin nopeasti ja on niin tuoretta, että sen avulla voidaan ohjata projektia. Mittaria voidaan siis verrata auton navigaattoriin, jonka pitää kertoa tulevasta käännöksestä ajoissa ja jatkuvasti; ei riitä, että navigaattori tai mittari kertoo, että käännöksen olisi pitänyt tapahtua kymmenen minuuttia sitten [Pfleeger, 1993]. Ketterien menetelmien adaptiivisuusperiaate edellyttää, että projektilla on jatkuvasti käytettävissä dataa sen suoriutumisesta, jota sitten käytetään suorituskyvyn parantamiseen. Lyhyempi palautesilmukka on parempi kuin pitkä,

eli on tärkeää, että palaute saadaan nopeasti ja usein, jolloin suuntaa voidaan korjata päättäväisesti ja oikeaan suuntaan.

Mittaamisen tulee olla taloudellista, eli sellaista, että mittaamisella saadut hyödyt ovat suuremmat kuin mittaamisen aiheuttamat lisäkustannukset. Tämä voi olla haastavaa, koska esimerkiksi Pfleegerin [1993] tutkimuksessa mittaaminen aiheutti 5-10% lisäkustannuksia. Helpoin tapa tehdä usein tapahtuvasta mittaamisesta taloudellista on automatisoida tiedonkeruu [Berard; Pfleeger, 1993].

Mittareiden tulee olla objektiivisia, eli että mittari antaa samat tulokset mittaajasta ja mitattavista riippumatta [Haikala ja Märijärvi, 2000]. Useimmiten ohjelmistokehitystä mitattaessa luotetaan ainakin osittain kyselyihin, vertais- tai alaisarvioihin, joten näitä mittareita käytettäessä on oltava erityisen huolellinen objektiivisuuden suhteen.

Luotettavuus tarkoittaa, että mikäli mitattava asia ei muutu, myös mittarin tulokset pysyvät samoina [Haikala ja Märijärvi, 2000].

Haikalan ja Märijärven [2000] mainitsema selektiivisyys tarkoittaa, että mittarin ja mitattavan asian välillä on johdonmukainen suhde. Jos mittari on selektiivinen, ovat mittarin ilmaisemat ongelmat kohdistettavissa mahdollisimman tarkasti ohjelmistokehitysprosessin osaan tai ohjelmistokehitystiimiin. [Haikala ja Märijärvi, 2000].

Mittareiden arvioimiseen tässä tutkielmassa käytetään lähinnä tosiaikaisuutta, taloudellisuutta ja objektiivisuutta. Relevanttius on arvovalinta ja kokonaisuus taas liittyy mittariyhdistelmään eikä vain yksittäiseen mittariin. Luotettavuus ja selektiivisyys ovat hieman vaikeita arvioida tuntematta projektin toimintaympäristöä.

6. Mittareita

Tässä luvussa esitellään joitakin mittareita ja kerrotaan niiden hyvistä ja huonoista puolista. Kuten aiemmissa luvuissa esiteltiin, mittareilla voidaan mitata monenlaisia asioita, ja niinpä tässä luvussa olevat mittarit on eritelty niiden mittaaman ominaisuuden mukaisesti. Ohjelmistoprojektin kannalta oleellisina mitattavina asioina pidetään tässä tuottavuutta, laatua, ja ketterien menetelmien omaksumista.

6.1. Tuottavuus

Tuottavuutta mittaavia mittareita ovat esimerkiksi tiimin nopeus (en. *velocity*), koodirivituotannon määrä, valmiiksi saatujen työlistan töiden määrä, versionhallintaan tehtyjen kooditoimitusten lukumäärä sekä näiden erilaiset muunnelmat.

Tiimin nopeus on asia, joka yleensä tulee ensimmäisenä mieleen ketterän prosessin mittaamista ajatelleen, sanasta itsestään kun voisi päätellä, että luku tarkoittaa nimenomaan tuotekehityksen nopeutta tai vauhtia. Nopeus tarkalleen ottaen kuitenkin tarkoittaa sitä, kuinka suureksi arvioimansa työmäärän tiimi saa valmiiksi sprintin aikana. Tiimin tehokkuuden nousun myötä arviot siten pienenevät, joten tehokkuusmittariksi nopeus ei ole sopiva. Nopeuden arvo on silti mittareidenkin kannalta hyödyllinen niin kuin kohdassa 8.6 näytetään.

Toinen yleisesti käytössä oleva tuottavuuden mittari on on tuotettujen koodirivien määrä. Jos ohjelmistotuotanto nähdään liukuhihnatyyppisenä työnä, niin silloin on loogista ajatella, että mitä enemmän koodia saadaan aikaan, sitä paremmin koneisto toimii. Todellisuudessa asia ei ole näin yksinkertainen. Ensiksikin, koodirivit eivät ole samanarvoisia. Eri ohjelmiston osat ovat tyypillisesti kompleksisuudeltaan aivan eri luokkaa, ja tehokas tiimi tai osaava koodari saa tehtäväkseen vaikeimmat tehtävät. Toiseksi, osaava ohjelmistosuunnittelija kirjoittaa koodia tyypillisesti huomattavasti tiiviimmin kuin aloitteleva. Kolmanneksi, ohjelmistokokonaisuuden refaktoroinnin myötä koodimassa voi monesti pienentyä, jolloin tehty työ ei näy.

Kompleksisuuden tuomaa koodirivien eriarvoisuutta voi yrittää tasapainottaa esimerkiksi käyttämällä kompleksisuuskerrointa, esimerkiksi Halsteadin mittaa tai McCaben syklomaattista kerrointa [Haikala ja Märijärvi, 2000]. Tällöin lasketaan paitsi koodimassan määrä myös analysoidaan sen monimutkaisuus, joka pisteytetään kertoimeksi tuottavuutta arvioitaessa.

Avuksi voi ottaa myös Boehmin kehittämän COCOMO-mallin [Haikala ja Märijärvi, 2000], joka on alunperin kehitetty työmääräarvioiden tekemiseen. Menetelmässä varsinaista arviota painotetaan ottamalla huomioon monia ohjelmiston toteuttamisen monimutkaisuuteen liittyviä seikkoja, kuten luotettavuusvaatimukset, suoritusaikavaatimukset, ohjelmoijien ja suunnittelijoiden kyvykkyys, sovellusalueen tuntemus ja kehitystyökalujen käyttö, ja näiden perusteella annetaan kerroin arvioidulle työmäärälle [Haikala ja Märijärvi, 2000].

Kolmas erityisesti ketterissä projekteissa käytettävissä oleva tuottavuuden mittari on valmiiksi saatujen työlistan töiden määrä. Iteraation päätteeksi lasketaan täysin valmiiksi saadut käyttäjäkertomukset, korjatut virheet ja muu työlistalta otettu työ yhteen. Koska työlistalla olevat työt ovat keskenään eri suuruisia, voi mittaria tarkentaa käyttämällä töistä etukäteen annettua työmääräarviota suhdelukuna.

Neljäs tuottavuuden mittari on projektin versionhallintaan tehtyjen kooditoimitusten määrä. Mittarissa piilee muutamia heikkouksia. Ongelmia voi ilmetä esimerkiksi silloin kun versionhallintaan toimitetaan uutta koodia, mutta sitä

joudutaan korjaamaan esimerkiksi testauksen jälkeen. Tällöin huonoa koodia tekemällä saa mittarin arvoa kasvatettua.

6.2. Laatu

Pressmanin [1997] mukaan ohjelmiston laadun mittarit voidaan karkeasti jaotella oikeellisuutta, ylläpidettävyyttä, yhtenäisyyttä ja käytettävyyttä mittaaviin mittareihin, joista tässä luvussa keskitytään lähinnä kahteen ensimmäiseen. Tuotetun ohjelmiston oikeellisuutta voidaan mitata esimerkiksi testauksen ja testiautomaation metriikoilla ja virheiden tai regression määrällä. Ylläpidettävyyttä voidaan mitata mittaamalla esimerkiksi mutkikkuusmittoja, tarkastusten määrää sekä ohjelmiston muuttamiseen kuluvaa aikaa.

Luvussa 2 esitellyn automaattisen testauksen ja yksikkötestien tuottama data on mittaamisen kannalta erityisen käyttökelpoista, koska se syntyy automaattisesti joka kerta, kun testit ajetaan, eli ketterässä ohjelmistokehityksessä ihanteellisesti aina, kun ohjelmakoodia muutetaan. Näin data on taloudellisesti ja tosiaikaisesti, useimmiten useita kertoja sprintin aikana, saatavilla. Varsinaisten testitulosten lisäksi yksikkötestauksen metriikoista käyttökelpoisia ovat ns. funktiokattavuus, päätöskattavuus sekä ehtokattavuus, jotka kuvaavat ohjelmistomoduleille laadittujen testien laajuutta.

Ohjelmiston laadusta kertoo myös siinä havaittujen virheiden määrä. Pressman [1997] esittää yleisimmäksi mittariksi virheiden määrän tuhatta ohjelmakoodiriviä kohden. Mittarin ongelmana on ensinnäkin tosiaikaisuus. Ohjelmiston virheet havataan tyypillisesti pitkän ajan kuluessa, osa jo yksikkötestauksessa ja osan huomaa vasta asiakas. Mittarin tuottamaa dataa ei siis ole mahdollista käyttää työn ohjaamiseen. Toiseksi, mittari ei ole selektiivinen – virheidenhallintajärjestelmissä ei tyypillisesti määritellä, mistä nimenomaisesta koodirivistä virhe johtui, joten virheen jäljittäminen takaisin tiimitasolle kustannustehokkaasti on vaikeaa.

Kaikkein virheiden sijaan on helpompaa seurata ohjelmistoa tehdessä syntyneitä virheitä käyttämällä regressiotestausta. Regressio tarkoittaa ilmiötä, jossa aiemmin viaton toiminnallisuus on jonkin uuden toiminnallisuuden myötä vikaantunut. Regressiotestaus puolestaan tarkoittaa normaalia laajemman testisetin suorittamista, jotta voidaan havaita, onko regressiota tapahtunut. Koska regressiotestaus on mahdollista suorittaa usein ja tarkoin valikoiduille koodi-toimituksille, on se luonteeltaan soveltuvampi laatumittariksi. Edellisessä kohdassa esiteltyä mittaria voi soveltaa tähän – regressiotestauksessa löydetyt virheet voi suhteuttaa kokonaiskoodin tai uuden koodin määrään.

Toinen mahdollisuus on tarkastella miten paljon koodia on katselmoitu. Katselmointien määrä on Haikalan ja Märijärven [2000] mielestä tehokkain ja halvin tapa löytää ohjelmistossa olevat virheet. Kuitenkin, koska koodi joko on

katselmoitu tai sitten ei ole, niin kovin luotettavaksi matalan tason mittariksi tästä ei ole.

Ylläpidettävyyden mittaaminen suoraan on haastavaa, ja tyypillisesti sen mittaamiseen suositaan jonkinlaisten välillisten asioiden mittaamista. Pressmanin [1997] mukaan yksinkertaisin ylläpidettävyyttä mittaava mittari on MTTC, mean-time-to-change, eli aika, joka kuluu ohjelmistoon tulevan muutoksen analysointiin, suunnitteluun, toteutukseen, testaukseen ja jakeluun. Mittarin suurin ongelma on tosiaikaisuus – arvo voidaan laskea vasta, kun jotakin komponenttia pitää muuttaa.

Ylläpidettävyyttä mittaa myös alunperin Hitachin esittelemä pilaantuneisuus-arvo, joka tarkoittaa löydettyjen virheiden korjaamiseen kuluvia kustannuksia ohjelmiston valmistumisen jälkeen [Pressman, 1997]. Mittarin ongelmat ovat samankaltaisia kuin virheiden määrän mittaamisessa – mittaria voi käyttää vasta projektin päätyttyä.

Ylläpidettävyyden mittaamiseen voidaan käyttää myös erilaisia mutkikkuusmittoja, joista klassisimmat lienevät Halsteadin kompleksisuusmitta ja McCaben syklomaattinen kerroin. Halsteadin kompleksisuusmitalla voidaan laskea ohjelmiston pituus, ohjelman taso, ohjelmiston teoreettinen minimipituus ja nykyinen suuruusluokka sekä jopa työmäärä ja potentiaalinen virheiden määrä, kun tunnetaan ohjelmistossa esiintyvien eri operaattorien sekä operandien ja kaikkien ohjelmistossa käytettyjen operaattorien ja operandien määrä [Pressman, 1997]. McCaben syklomaattinen kerroin kertoo, kuinka monta mahdollista suoritusvuota koodilla on ja se lasketaan tyypillisesti funktiokohtaisesti [Haikala ja Märijärvi, 2000]. Mutkikkuusmitoista McCaben syklomaattinen kerroin on selvästi käytetympi [Haikala ja Märijärvi, 2000], ja Pressmanin [1997] mukaan se korreloi vahvasti koodissa olevien virheiden ja niiden korjaamiseen kuluvan ajan pituuden kanssa – joskin tämä tulos on varsin kiistanalainen. Koska mutkikkuusmittojen laskeminen voidaan automatisoida täysin ja laskeminen voidaan suorittaa koska tahansa, ne ovat taloudellisia ja tosiaikaisia mittareita. Tyypillisesti tämänkaltaiset metriikat lasketaan jatkuvan integroinnin yhteydessä, eli ne päivittyvät aina, kun koodia on muutettu.

6.3. Ketteryys

Cohn [2009] pitää kysymystä “miten ketteriä olemme” jopa melko esoteerisena. Hän painottaakin ketteryyden mittaamisessa mieluummin projektin etenemisnopeuden ja laadun mittaamista ja olettaa, että nämä seuraavat lisääntyneestä ketteryydestä. Itse ketteryydenkin mittaaminen on silti mahdollista ja jopa melko yleistä. Ketteryydellä tarkoitetaan yleisesti lähinnä luvussa 2 esiteltyjen ketterien työtapojen yleisyyttä organisaatiossa. Työtapoja voi arvioida tarkkailemalla työntekoa tai teettämällä projektin henkilöstöllä ja sidosryhmillä kyselyi-

tä työtavoista. Olemassa olevia kyselyjä ovat esimerkiksi Nokia Test, Krebsin Shodan Adherence Survey, Agile Evaluation Framework sekä Comparative Agility Assessment [Cohn, 2009; Sutherland, 2010].

Krebsin Shodan Adherence Surveyssä vastataan viiteentoista kysymykseen asteikolla yhdestä kymmeneen. Jokainen kysymys esittelee jonkin XP-menetelmän periaatteen tai käytännön sekä kriteerit, jotka sen yhteydessä tulisi täyttää [Cohn, 2009]. Krebsin ja Krollin Agile Evaluation Framework on vastaavanlainen, mutta nimensä mukaisesti se on ikään kuin kyselykehys – se ei määrittele kysyttäviä kysymyksiä, vaan ainoastaan antaa suositukset niiden määrästä ja luonteesta [Cohn, 2009].

Cohn [2009] esittelee lisäksi itse kehittelemänsä ns. Comparative Agility Assessmentin, jonka tarkoitus on verrata organisaatio ketteryyttä kilpailijoiden saavuttamaan ketteryystasoon. Kysely on ilmainen ja internet-pohjainen. Kysely on suunniteltu siten, että periaatteessa ScrumMaster yksinkin voi täyttää kyselyn melko luotettavasti tiiminsä puolesta. Kysely lähestyy ketteryyttä seitsemän ulottuvuuden kautta. Ne ovat tiimityö, vaatimukset, suunnittelu, tekniset käytännöt, laatu, kulttuuri ja tietämyksen luominen. Koska kyselyn tarkoitus on vertailla organisaatiota toisiin, on sen käyttökelpoisuus yksittäisen organisaation mittarina kyseenalainen. Comparative Agility Assessment on myös hyvin raskas sisältäen 125 kysymystä.

Kenties tunnetuin ketteryyskyselyistä on ns. Nokia Test, jonka alunperin on kehitellyt Bas Vodde Nokia Networksilla, ja jonka pisteytettäväksi mittaristoksi on jatkojalostanut Jeff Sutherland. Testi pisteyttää Scrumin käyttöönottoa organisaatiossa. Nokia Test kysyy kysymyksiä mm. iteraatioista, testauksesta, tuotteen työlistasta, Scrumin eri rooleista, työmääräarvioista, burndown-käyristä ja tiimistä. Tulos on arvosana asteikolla nollasta yhdeksäänkymmeneen. Testi sisältää objektiivisia kysymyksiä (esimerkiksi työlistan olemassaolosta) ja subjektiivisia kysymyksiä (esimerkiksi tiimin toiminnasta) [Sutherland, 2010]. Koska osa kysymyksistä on subjektiivisia ja vastaajan näkemyksestä riippuen niihin voidaan siis saada hyvinkin erilaisia vastauksia, olisi mittaus parasta tehdä koko organisaatiolle tehtävänä kyselynä. Tämä kuitenkin tekee mittarista hyvin kalliin. Organisaation ketteryys tuskin vaihtelee lyhyen ajan kuluessa, joten todennäköisesti esimerkiksi puolivuosittain järjestetyt kyselyt tarjoavat riittävän tosiaikaista dataa, jolloin mittarin kalleus jää suhteellisesti pienemmäksi.

Kohdassa 6.1. esitelty nopeus ei siis ollut hyvä mittari tuottavuuden mittaamiseen, mutta organisaation omaksuman ketteryyden mittaamiseen se on parempi. Nopeuden ei pidä kasvaa eikä suurempi nopeus ole parempi, vaan hyvin toimivassa ketterässä tiimissä nopeus pysyy tasaisena sprintistä toiseen. Nopeuden pysyessä tasaisena tiimi saa eri sprinteissä aikaan samansuuruiseksi

arvioimansa työmäärän. Tämä tarkoittaa sitä, että tiimi osaa arvioida tekemisensä hyvin, ja että tiimi saa tehtyä työtänsä ilman ulkoisia häiriöitä. Mittari on hyvin yksinkertainen, joten se täyttää Cohnin [2009] tärkeimmän kriteerin. Scrum-menetelmässä dataa nopeudesta tuotetaan automaattisesti, joten mittarina se on hyvin kevyt toteuttaa. Se on myös hyvin tosiaikainen, koska sen arvo päivittyy jokaisen sprintin päätteeksi. Sen sijaan mittari ei ole kovin selektiivinen – nopeuden tasaisuuteen vaikuttavat hyvin monet ja hyvin erilaiset asiat.

Toinen tiimin arviointikykyä tai työrauhaa ja siten saavutettua ketteryyden tasoa mittaava mittari on Cohnin [2009] esittelemä sprintin aikana sprintin työlistalta pudotettujen töiden määrä. Mittari tarkoittaa työn määrää, jonka tiimi on suunnittelutapaamisessa arvioinut voivansa tehdä sprintin aikana, mutta joka on syystä tai toisesta jäänyt tekemättä. Karkeasti ottaen, jos tiimi on arvioinut työmäärän oikein, on se jäänyt tekemättä jonkin ulkopuolisen häiriön takia. Jos taas ulkopuolista häiriötä ei ole ollut, tiimi on arvioinut työt väärin. Mittarin arvo päivittyy joka sprintin jälkeen ja se on helppo saada sprintin katselmointipalaverin jälkeen, joten mittari on taloudellinen ja tosiaikainen. Mittarin luotettavuus ei kuitenkaan ole aivan selvä, koska ohjelmistotyö on monesti arvaamatonta, ja työ saattaa häiriintyä myös muista kuin ihmisistä johtuvista syistä.

7. Yhteenveto

Ohjelmistoprojektin mittaaminen on hyvin vaikeaa. Tuskin mikään mittareista on sellainen, jonka antamien tulosten relevanttiudesta ei voisi kiistellä. Ja on vaikea löytää mittaria, jota ei vähintään muutamalla tavalla voisi huijata. Eikä taida olla olemassa ohjelmistosuunnittelijaa, jolla ei olisi huonoja kokemuksia mittareiden käytöstä.

Toisaalta nykyhetken tarkkanäköinen analyysi ja parannusten etsiminen adaptiivisesti ovat ketterien menetelmien ytimessä, ja on vaikea nähdä, miten tehtyjä muutoksia voi arvioida ilman mittaamista. Mittarit ovat parhaimmillaan silloin kun motivoituneet ja suorituksensa sekä ohjelmistonsa kehittämisestä kiinnostuneet ohjelmistosuunnittelijat näkevät ne sellaisina kuin niiden pitäisi olla – ohjelmistoprojektin ”navigaattorina”. Tällaisessa tilanteessa huijaaminen vahingoittaa ainoastaan tiimiä itseään, ja mittari on tärkeä indikaattori, jolla varmistetaan uudistusten aikaansaamat parannukset laadussa, tuottavuudessa ja ketteryydessä.

Viiteluettelo

[Adolph, 2006] Steve Adolph, "What lessons can the agile community learn from a maverick fighter pilot?", In: *Proceedings of AGILE 2006 Conference* (2006), 94-99.

[Agile, 2010] Version One, State of agile survey, 2010. Available as http://www.versionone.com/pdf/2010_State_of_Agile_Development_Survey_Results.pdf. Checked May 30, 2011.

[Augustine et al., 2005] Sanjiv Augustine, Bob Payne, Fred Sencindiver and Susan Woodcock, Agile Project Management: Steering from the edges. *Communications of the ACM* **48**, 12 (Dec. 2005), 85-89.

[Beck et al., 2001] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland and Dave Thomas, Agile manifesto. 2001. Available as http://agilemanifesto.org/. Checked May 30, 2011.

[Berard] Edward V. Berard, Metrics for Object-Oriented Software Engineering. Available as http://www.ipipan.gda.pl/~marek/objects/TOA/moose.html. Checked May 30, 2011.

[Cohn, 2009] Mike Cohn, *Succeeding with Agile*. Addison-Wesley, 2009.

[Haikala ja Märijärvi, 2000] Ilkka Haikala ja Jukka Märijärvi, *Ohjelmistotuotanto*. Satku, 2000.

[Hughes and Cotterell, 2006] Bob Hughes and Mike Cotterell, *Software Project Management*. McGraw-Hill, 2006.

[Larman, 2004] Craig Larman, *Agile and Iterative Development, a Manager's Guide*. Addison-Wesley, 2004.

[Lewis, 1998] James P. Lewis, *Mastering Project Management*. McGraw-Hill, 1998.

[Pfleeger, 1993] Shari Lawrence Pfleeger, Lessons learned in building a corporate metrics program. *IEEE Software* **10**, 3 (May 1993), 67-74.

[Pressman, 1997] Roger S. Pressman, *Software Engineering, a Practitioner's Approach*, Fourth Ed., McGraw-Hill, 1998.

[Schwaber and Beedle, 2002] Ken Schwaber and Mike Beedle, *Agile Software Development with scrum*, Prentice-Hall Inc., 2002.

[Schwaber, 2004] Ken Schwaber, *Agile Project Management with Scrum*. Microsoft Press, 2004

[Scrum Guide] Scrum Alliance, Scrum guide. Available as http://scrumcommunity.pbworks.com/f/scrum_guide.pdf. Checked May 30, 2011.

[Sutherland, 2010] Jeff Sutherland, ScrumButt test, 2009. Available as http://jeffsutherland.com/scrumbutttest.pdf. Checked May 30, 2011.

[Valett and McGarry, 1989] J. Valett and F. E. McGarry, A summary of software measurement experiences in the software engineering laboratory. *Journal of Systems and Software* **9**, 2 (Feb. 1989), 137-148.

E-kirjat ja niiden lukemiseen tarkoitettujen sovellusten käytettävyys tablet-laitteella

Sini Tistelgrén

Tiivistelmä.

Tässä tutkielmassa käsitellään elektronisten kirjojen käytettävyyttä tablet-laitteella, erityisesti Applen iPad-laitteella. Olen rajannut tutkielmani koskemaan nimenomaan Pohjois-Amerikan ja Suomen e-kirjamarkkinoita. Pohjois-Amerikassa e-kirja on jo yleisessä käytössä, mutta Suomessa vasta tulollaan. Rajaan e-kirjoja koskevan kielikysymyksen tutkielmastani pois, sillä tarkoitus on keskittyä käytettävyyteen, eikä niinkään käyttökokemukseen tai e-kirjan leviämiseen. E-kirjat ovat jatkuvasti yleistyneet, mutta niiden käytettävyyttä on tutkittu vielä varsin vähän. Tarkoituksenani on määritellä e-kirjojen käytettävyyden kriteerit. Tutkielman alussa perehdytään lyhyesti e-kirjojen historiaan, erilaisiin e-kirjojen lukulaitteisiin, näyttötekniikkaan ja tiedostomuotoihin. Tuon myös esiin e-kirjojen hyvät ja huonot puolet. Tämän jälkeen pohdin käytettävyyttä ja e-kirjojen käytettävyyskriteerien määrittelyä. Tutkielman loppupuolella arvioin Elisa Kirja -palvelun käytettävyyttä.

Avainsanat ja -sanonnat: sähkökirja, e-kirja, käytettävyys, e-lukija, tablet-laite, iPad, Elisa Kirja

CR-luokat: J.7, H.5.2

1. Johdanto

Digitaalisen kirjallisuuden myynti ohitti ensimmäistä kertaa painetun kirjallisuuden myynnin Yhdysvalloissa helmikuussa 2011. Kasvua vuoden takaiseen oli hurjat 202 %. [Pepitone, 2011]

TNS Gallupin vuosittain Helsingin kirjamessuilla tekemä kyselytutkimus paljastaa, että kiinnostus e-kirjoihin on kasvussa Suomessakin. Vuonna 2010 e-kirjaa oli lukenut 15 % vastaajista, mutta lukulaitteen omisti vain pari prosenttia. Vuoden 2011 tutkimuksessa 20 % vastaajista oli lukenut e-kirjaa ja e-kirjan lukulaite oli 6 %:lla vastaajista. Yli puolet kyselyyn vastanneista uskoi e-kirjan nousevan tulevaisuudessa painetun kirjan rinnalle. Kuitenkin vain 3 % vastaajista uskoi e-kirjan syrjäyttävän perinteisen painetun kirjan. Tutkimuksessa haastateltiin 300 kirjamessuilla kävijää [Peltoniemi, 2011].

E-kirjojen menestystä on hidastanut lukulaitteiden kalleus ja paikoin e-kirjojen heikko saatavuus. Muutaman viime vuoden aikana markkinoille on kuitenkin ilmestynyt useilta valmistajilta uusia ja aikaisempaa edullisempia lukulaitteita sekä tablet-laitteita, jotka ovat tuoneet e-kirjat yhä useamman kuluttajan ulottuville. Suomessa monet perinteisiäkin kirjoja myyvät verkkokaupat, kuten Suomalainen kirjakauppa ja Akateeminen kirjakauppa, ovat ottaneet e-kirjat valikoimiinsa. Markkinoille on ilmestynyt myös palveluita, jotka ovat keskittyneet tarjoamaan pelkästään e-kirjoja kuluttajille, tällainen on esimerkiksi Elisa Kirja. Lisäksi kirjastot ovat yhä aktiivisemmin alkaneet tarjota e-kirjoja ja e-lukijoita käyttäjien lainattaviksi.

2. Mikä e-kirja?

E-kirja on digitaalisessa muodossa oleva kirjallinen teos, joka on luettavissa useilla erilaisilla elektronisilla laitteilla. E-kirja tunnetaan myös nimillä elektroninen kirja, sähköinen kirja, sähkökirja ja digitaalinen kirja. Suomenkielinen terminologia ei vielä ole täysin vakiintunut. Oxford Dictionary of English määrittelee e-kirjan olevan ”elektroninen versio painetusta kirjasta” [Oxford Dictionaries, 2010]. E-kirja voi kuitenkin olla olemassa myös itsenään, ilman painettua teosta.

2.1 Elektronisen kirjan historia

E-kirjojen historian voidaan katsoa alkavan vuodesta 1971, jolloin Michael Hart käynnisti Gutenberg-projektin. Projekti nimettiin kirjapainotekniikan pioneerin Johannes Gutenbergin mukaan, ja sen tavoitteena oli tarjota kirjallisia teoksia ilmaiseksi digitaalisessa muodossa ja vieläpä kaikkialla maailmassa. Nykymittapuulla tavoite kuulostaa melko vaatimattomalta, mutta ajankohdan ja teknisen kehityksen huomioon ottaen Hartin projekti oli erittäin kunnianhimoinen [Wikipedia, 2011a].

Elokuussa 1989 Gutenberg-projektissa oli 10 e-kirjaa, mutta kymmenen vuotta myöhemmin e-kirjoja oli jo yli 2000. Vuoden 2009 helmikuussa Gutenberg-projekti oli tuottanut yli 32 500 digitalisoitua kirjallista teosta [Lebert, 2009]. E-kirjojen alkutaival kulki pitkälti käsi kädessä Internetin alkutaipaleen kanssa, muutama sananen siis Internetin kehittymisestä.

Internet ei ollut olemassa nykyisessä muodossaan vielä Gutenberg-projektin alkuaikoina. Internetin kehittyminen vaatimattomasta Pentagonin käyttöön luodusta verkosta tavallisten käyttäjien saatavilla olevaksi maailmanlaajuiseksi tietoverkoksi (alkaen 1994) vei kauan, mutta tällä prosessilla on ollut suuri vaikutus e-kirjojen yleistymiseen. [Lebert,2009].

Internetin taival alkoi Pentagonin käyttöön luotuna ARPAnet (Advanced Research Project Agency) verkkona vuonna 1969. Bob Kahnin ja Vinton Cerfin

kehittelemä TCP/IP-protokolla tuli mukaan vuonna 1974 ja ARPAnet laajeni kattamaan Yhdysvaltain hallinnolliset virastot, tutkimuskeskukset ja yliopistot. ARPAnetin levittyä laajemmin akateemiseen käyttöön alettiin verkkoa kutsua Internetiksi. Yhdysvaltain puolustusvoimien verkko puolestaan kehittyi tämän jälkeen ARPAnetistä erillään omana MilNet-tietoverkkonaan. Tim Berners-Lee keksi CERNissä varsinaisen webin vuosina 1989–1990. Ensimmäinen selain (Mosaic) julkaistiin marraskuussa 1993 ja Internet alkoi levitä tavallistenkin käyttäjien saataville, kuitenkin aluksi lähinnä Pohjois-Amerikassa [Lebert, 2009]. Suomessa Internetin leviäminen alkoi EUnet Finlandin alkaessa tarjota Internet-yhteyksiä yritys- ja henkilöasiakkaille vuonna 1993 [Wikipedia, 2011b].

Internetin kehittyminen toi aivan uudenlaiset tiedonsiirtomahdollisuudet ja näin ollen myös e-kirjojen levittäminen muuttui yhä helpommaksi, ja uusi nopea tiedonsiirto toi e-kirjat yhä useampien saataville ja kiinnostus niitä kohtaan kasvoi voimakkaasti [Lebert, 2009].

Amazon.com alkoi myydä painettuja kirjoja Internetissä vuonna 1995. Internetin käyttäjien määrä kasvoi räjähdysmäisesti ja vuonna 1997 käyttäjiä oli yli 100 miljoonaa ja lisää tuli noin miljoonan käyttäjän kuukausivauhdilla. Kirjastot Yhdysvalloissa toivat e-kirjat (kyse enimmäkseen tietokirjallisista teoksista) Internet-sivuillensa vuonna 1998, ja kustantajat alkoivat julkaista yhä enemmän kirjallisuutta Internetissä tekijöiden luvalla. Joulukuussa vuonna 2000 internetillä oli yli 300 miljoonaa käyttäjää. Vuonna 2003 e-kirjoja alettiin myydä Internetissä maailmanlaajuisesti [Lebert, 2009].

Merkkipaaluja e-kirjojen (ja Internetin) kehityksessä

1969 Pentagonin sisäinen verkko ARPAnet kehitellään

1971 Michael Hart käynnistää Gutenberg-projektin

1974 TCP/IP protokolla kehittyy ja ARPAnet laajenee kattamaan Yhdysvaltain hallinnolliset virastot, tutkimuskeskukset ja yliopistot

1989 Gutenberg-projekti saanut valmiiksi 10 e-kirjaa

1989–1990 Tim Berners-Lee kehittää Webin CERNissä

1993 Ensimmäinen selainohjelma Mosaic julkaistaan ja EUnet Finland tarjoaa Internet yhteyksiä ensimmäisenä Suomessa

1994 Internet leviää tavallisten käyttäjien saataville ja yleistyy lähinnä Pohjois-Amerikassa ja samaan aikaan ensimmäiset kustantamot siirtyvät ”digitaalisiksi”

1995 Amazon.com alkaa myydä painettuja kirjoja verkkokaupassa

1997 Internetillä on yli 100 miljoonaa käyttäjää

1998 Kirjastot Yhdysvalloissa alkavat tarjota Internet sivuillaan joitakin teoksia (tietokirjoja) digitaalisessa muodossa netLibraryn kautta

1999 Gutenberg-projektissa yli 2000 elektronista kirjaa

2000 Internetillä yli 300 miljoonaa käyttäjää

2003 E-kirjoja myydään maailmanlaajuisesti ja kirjastot Yhdysvalloissa tarjoavat lainattavaksi elektronisia versioita lähes kaikista suosikkiteoksista.

2.2 E-Kirjojen edut

E-kirjoilla on monia etuja verrattuna painettuihin kirjoihin. E-kirjoihin ei kulu paperia tai mustetta. E-kirjat eivät vie fyysistä tilaa kassissa tai kirjahyllyssä, ainoastaan muutamia megatavuja lukulaitteen tallennuskapasiteettia, jolloin e-kirjojen lukumäärän rajoittavaksi tekijäksi muodostuu ainoastaan lukulaitteen tallennuskapasiteetin määrä.

Esimerkiksi Applen iPad-laitteeseen mahtuu useita tuhansia e-kirjoja, riippuen mallin tallennuskapasiteetista (16-64gt). Tuhannen painetun kirjan mukana kantaminen olisi sula mahdottomuus; moni kuitenkin kuljettaa tablettia tai e-lukijaa mukanaan lähes kaikkialle ja samalla mukana kulkevat laitteeseen tallennetut e-kirjat.

Painetun kirjan hankkimista varten käyttäjä joutuu kulkemaan kirjakauppaan tai kirjastoon. Kirjoja verkkokaupasta hankkiessa täytyy odotella postitse saapuvan teoksen saapumista. E-kirjan hankkiminen ei kuitenkaan ole sidottu paikkaan tai aikaan, vaan pääsääntöisesti vaatimuksena on pelkkä Internet-yhteys lukulaitteesta ja e-kirjan saa ostettua tai lainattua ja käyttöönsä välittömästi. E-kirjojen käyttöön liittyy myös ekologinen näkökulma, sillä painetun kirjan valmistamiseen kuluu yli kolme kertaa enemmän raakamateriaaleja ja jopa yli 78 kertaa enemmän vettä kuin esimerkiksi iPad-laitteen valmistamiseen [Goleman and Norris, 2010].

E-lukulaite on usein kallis kertainvestointi, mutta e-kirjat ovat monesti hieman halvempia kuin painetut kirjat. Lisäksi e-kirjoja on saatavilla runsaasti ilmaiseksi (vrt. Gutenberg-projekti tai esimerkiksi Elisa Kirja -palvelun ilmaiset klassikot). Myös monet kirjastot erityisesti Pohjois-Amerikassa tarjoavat nykyisin e-kirjoja lainattaviksi, trendi rantautunee Suomeenkin pikku hiljaa.

E-kirjoista on monesti mahdollista tehdä varmuuskopio riippuen e-kirjan tiedostotyypin suojauksesta (DRM). Mikäli lukulaite rikkoutuu tai kaikki data menetetään, tarjoavat eräät palveluntarjoajat (kuten Amazon.com ja Barnes&Noble.com) käyttäjälle ilmaiseksi uuden e-kirjan menetetyn tilalle pilvipalveluidensa kautta [Wikipedia,2011a]. Myös Apple toi iPadiin iCloud-pilvipalvelun iOS 5 -päivityksen myötä. IPad-laite on mahdollista varmuuskopioida iCloudiin suoraan wlan-verkon yli [Apple, 2011].

Myös kirjailijoiden näkökulmasta tarkasteltuna e-kirjoilla on omat etunsa. Kirjailijoiden on helpompi, halvempi ja nopeampi julkaista teoksiaan e-kirjoina kuin painettuina kirjoina [My Ebook Design, 2011].

Monet nykyaikaiset e-kirjojen lukulaitteet tarjoavat erilaisia helppokäyttötoimintoja, mm. mahdollisuuden muuttaa fonttien kokoja, tehdä kirjanmerkkejä ja merkintöjä tai tiedonhakuja suoraan kirjasta. Lisäksi jotkut laitteet tarjoavat teksti puheeksi -toimintoa, joka on hyödyksi varsinkin sokeille tai näkövammaisille käyttäjille [Jamali et al., 2008; Cleary et al., 2011].

2.3 E-kirjojen haitat

Varhaisimmissa lukulaitteissa näyttötekniikka oli huono ja luettavuus kaukana fyysisen kirjan luettavuudesta. Laitteet olivat painavia ja epämiellyttävän tuntuisia kädessä. Lisäksi laitteet olivat kalliita ja sisältöä ei ollut saatavilla kovinkaan paljoa [Heikkilä et al., 2011].

Lisäksi useat tutkimukset paljastivat aiempien laitteiden käytettävyyden olleen varsin huono. Käyttäjät kokivat erityisesti navigaation toimineen huonosti [Henke, 2002; Schcolnik, 2001; Landoni and Wilson, 2002].

Osa edellä mainituista ensimmäisen sukupolven lukulaitteiden ongelmista on läsnä nykypäivänäkin. Laitteet ovat edelleenkin kalliita ja tarjontaa on paikoin rajallisesti (esim. suomen kielellä). Laitteiden korkea hinta rajaa maapallon köyhimmät väestöryhmät suoraan ulos potentiaalisten käyttäjien joukosta. Hinnan ohella laitteiden lataamiseen tarvitsema sähkövirta asettaa omat rajoitteensa. Tavallista kirjaa voi sen sijaan lukea autiolla saarellakin. Vaikka laitteen lataaminen olisi aina mahdollista, alkaa laitteen akku huonontua ajan myötä käytössä ja joskus tulevaisuudessa se lakkaa toimimasta. Lisäksi uudetkin e-lukemiseen tarkoitetut laitteet kärsivät edelleen käytettävyysongelmista edeltäjiensä tavoin.

Lisäksi e-kirjojen DRM (Digital Rights Management) suojaukset aiheuttavat päänvaivaa. DRM-suojatun sisällön omistusoikeus ja yhteensopivuus tulevaisuudessa hankittavien laitteiden kanssa askarruttaa monia [Li, 2008]. Lisäksi useat eri tiedostomuodot ja niiden yhteensopivuus erilaisissa laitteissa eivät ole kovinkaan selviä. Tuntuu kohtuuttomalta ajatukselta investoida vaikkapa 200€ laitteeseen ja toinen 200€ e-kirjojen ostamiseen, jos kirjat eivät olekaan käytettävissä seuraavalla lukulaitteella, jonka hankinta tapahtuu joskus tulevaisuudessa.

Elektronisia laitteita ei ole tehty kestämään ikuisesti ja tarve uusia laitteistoa tulee vastaan väistämättä joskus. Lisäksi e-lukija kärsii lähes kaikista muistakin elektronisia laitteita koskevista rajoitteista. Kosteus, suuret lämpötilanvaihtelut ja tipahtaminen tai voimakas tärähdys saattavat riittää hajottamaan laitteen. Toisinaan laitteet vikaantuvat ilman mitään näkyvää syytä ja huollon jälkeen yleensä kaikki data on menetetty. Tavallisen kirjan voi puolestaan surutta tiputtaa korkealtakin, ja se selviää niin pakkasesta kuin kuumuudesta tiettyyn rajaan asti. Tavallinen kirja ei myöskään mene epäkuntoon.

Fyysisen teoksen eteenpäin myyminen ja lainaaminen ovat itsestään selviä asioita. Elektronisten kirjojen kohdalla myynti ei onnistu (ellei myy laitetta), ja henkilöasiakkaiden keskinäinen kirjojen lainaaminen on mahdollista kirjoitushetkellä vain Barnes&Noblen Nook-lukulaitteen ja lukusovelluksien tarjoamalla LendMe-palvelulla ja Amazonin Kindle-laitteilla. Ongelmia lainaamiseen aiheuttaa myös se, että kustantamot eivät ole tehneet kaikista e-kirjoista lainkaan lainattavia tai kunkin teoksen eteenpäin lainaaminen onnistuu vain esimerkiksi kerran. Itse lainausprosessi on myös varsin työläs ja vaatii toimenpiteitä sekä kirjan omistajalta että lainaajalta [Woo, 2011].

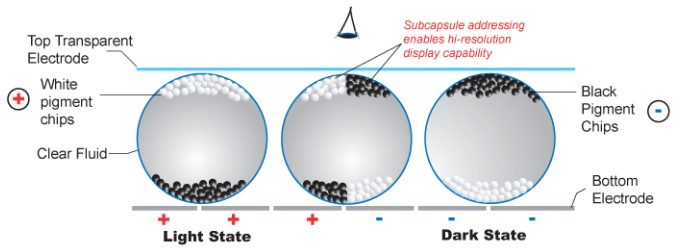
Tavallisen kirjan voi antaa käärepaperiin paketoituna vaikkapa joululahjaksi. Joku saattaa myös haluta sisustaa kirjoilla. Vaikka toiselle voikin antaa lahjaksi lahjakortin verkkokirjakauppaan tai e-kirjan lukijan (ja sen sisältämät kirjat) voi laittaa kirjahyllyyn, ei vaikutelma ole sama. Elektronisella kirjalla ei voida mitenkään täysin simuloida fyysisen kirjan olemusta ja sen käyttäjille tuomaa tunnetta. Eikä näin ole tarkoituskaan.

2.4 E-kirjojen lukulaitteet

E-kirjojen lukemiseen tarkoitetut laitteet voidaan ryhmitellä kahteen kategoriaan: erityisesti e-kirjojen lukemista varten suunniteltuihin e-lukijoihin ja ns. ”monikäyttöisiin laitteisiin”. Monikäyttöisiin laitteisiin kuuluvat esimerkiksi tietokoneet, älypuhelimet ja tablet-laitteet. Monikäyttöisiä laitteita voidaan käyttää lukemiseen, joko laitteessa valmiiksi olemassa olevan ohjelmiston avulla tai lataamalla laitteeseen kolmannen osapuolen lukuohjelma [Groner et al., 2010].

E-lukijat hyödyntävät elektroforeesiin perustuvaa näyttötekniikkaa, jota kutsutaan yleisesti e-ink- tai e-paper-tekniikaksi. E-ink-näytöt pyrkivät ominaisuuksiltaan mahdollisimman lähelle oikeaa paperia. E-ink-näytöt tarjoavat laajan katselukulman, mahdollistavat lukemisen kirkkaassakin valossa ja kuluttavat erittäin vähän virtaa. Alhainen virrankulutus johtuu elektroforeesiin pohjautuvasta tekniikasta, jossa näyttö kuluttaa virtaa ainoastaan, kun näytössä olevaa kuvaa muutetaan (kuva 1). E-ink-näytöltä lukiessa virtaa kuluu siis vain sivuja kääntäessä. Monet valmistajat lupaavat e-lukijoilleen huikeita käyttöaikoja (jopa 30 päivää).

E-ink-näyttö ei ole taustavalaistu, joten lukemiseen tarvitaan normaalin valaistuksen. Monet käyttäjät ovat kokeneet e-ink-näytöiltä lukemisen silmiä vähemmän rasittavana kuin lcd- tai led-näytöltä lukemisen. E-ink-näyttöjen huono puoli on hidas vasteaika. Sivujen kääntäminen ei ole yhtä ripeää kuin esim. tabletilla tai mobiililaitteella lukiessa [Pratt, 2010].



Kuva 1. E-ink-näyttötekniikan toimintaperiaate [E Ink, 2010]

Tällä hetkellä suurin osa markkinoilla olevista e-lukulaitteista käyttää mustavalkoisia e-ink-näyttöjä ja tämä tuo sisällön suhteen omat rajoitteensa, joista esimerkiksi tablet-laitteet eivät kärsi. Markkinoille on kuitenkin tullut viime kuukausina muutama värinäytöllinen e-lukija, jolloin tablettien etumatka tällä saralla kaventuu [Savov, 2011].

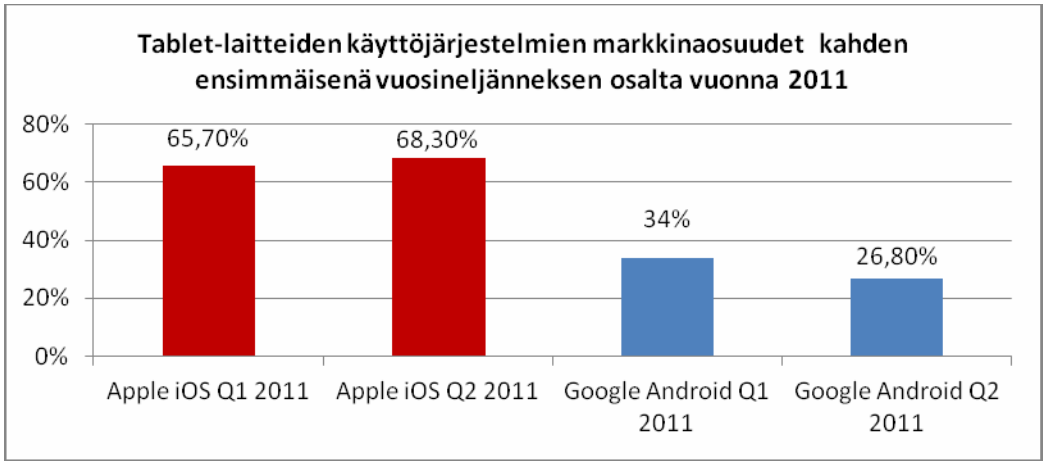
E-ink-näytön tuoma lukumukavuus ja pitkä akkukesto ovat selkeästi e-lukijoiden etuja. Monesti e-kirjanlukijat ovat myös kevyempiä ja pienempiä kuin tablet-laitteet. Selkeä negatiivinen seikka kuitenkin on, että e-lukija on suunniteltu vain lukemista varten, jolloin käyttäjä ei voi hyödyntää laitetta juuri muuhun, toisin kuin tablet-laitteita. Kuvassa 2 on esimerkkejä markkinoilla olevista e-lukijoista.

Kuva 2. Esimerkkejä markkinoilla olevista e-lukijoista [Topten reviews, 2011]

Monikäyttöisiin lukulaitteisiin lukeutuvat tietokoneet, älypuhelimet ja tablet-laitteet. Näillä laitteilla voidaan lukea e-kirjoja erillisten kolmansien osapuolten sovellusten kautta [Groner et al., 2010].

Tarkoituksenani on tutkia e-kirjojen käytettävyyttä nimenomaan tabletilla (Applen iPad), joten en pureudu sen tarkemmin muihin monikäyttöisiin lukulaitteisiin. Tabletti tunnetaan myös nimillä taulutietokone, paneelitietokone tai sormitietokone. Tabletleissa ei ole fyysistä näppäimistöä vaan kosketusnäyttö, jota ohjataan sormin ja toisinaan myös osoitinkynällä eli styluksella. Tyypillisimmin tablettien koko on 7-10 tuumaa. Tablet-laitteita käytetään usein tilanteissa, joissa tarvitaan kevyt, pieneen tilaan mahtuva laite, joka kulkee helposti mukana. Esimerkiksi opiskelijat ja paljon matkustavat ovat ottaneet laitteen avosylin vastaan [Wikipedia, 2011b]. Enemmistö tällä hetkellä markkinoilla

olevista tablet-laitteista käyttää Applen iOS- tai Googlen Android-käyttöjärjestelmää (vrt. taulukko 1) [IDC,2011].



Taulukko 1. Markkinaosuudet tablet-laitteiden käyttöjärjestelmissä [IDC, 2011]

Tablettien suurin ero e-lukijoihin verrattuna (monikäyttöisyyden lisäksi) on niissä käytetty näyttötekniikka. Tablet-laitteet käyttävät taustavalaistuja lcd-näyttöjä, esimerkiksi iPadissa on led-taustavalaistu kapasitiivinen kosketusnäyttö. Käyttäjät ovat raportoineet taustavalaistuilta näytöiltä lukemisen rasittavan silmiä enemmän kuin e-ink-näyttöjen. Lisäksi käyttäjät kertovat taustavalaistujen näyttöjen kärsivän e-ink-näyttöjä useammin erilaisista valaistuksen aiheuttamista heijastuksista [Cleary et al., 2011]. Toisaalta Reed Collegessa tehdyssä tutkimuksessa [Marmarelli and Ringle, 2010] (johon osallistuivat Reed Collegen erääseen valtio-opin seminaariin syksyllä 2010 osaa ottaneet opiskelijat), käyttäjistä vain yksi kertoi silmien rasittuvan iPadin näytön katselusta. Muut tutkimukseen osanottajat raportoivat positiivisista kokemuksista, eivätkä havainneet tekstin luettavuudessa juurikaan eroja iPadin ja e-ink-tekniikkaa käyttävään Amazon Kindle DX -laitteen välillä.

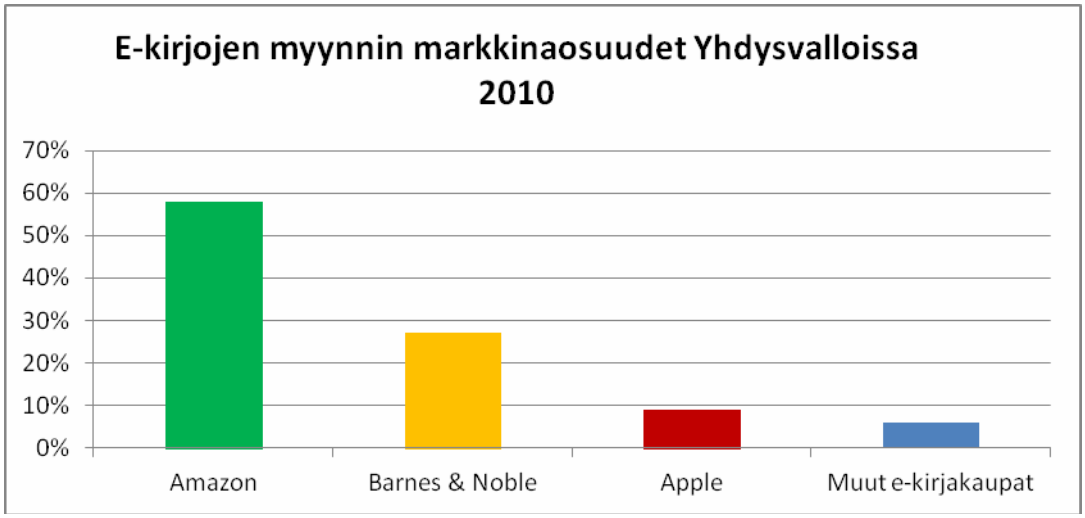
Tablettien etuna on niiden monipuolisuus; ne eivät ole pelkkiä e-kirjojen lukulaitteita vaan henkilökohtaisia kaikkialle mukana kulkevia tietokoneita, joilla on mahdollista surfata Internetissä, lukea sähköpostit, katsoa videoita, kuunnella musiikkia, pelata ja tehdä muistiinpanoja. Tableteille on saatavilla myös paljon erilaista sisältöä ohjelmistoalustojen tarjoamista sovelluskaupoista [Cleary et al., 2011].

2.5 E-kirjojen tiedostomuodot

E-kirjamarkkinoilla on tällä hetkellä tarjolla useita erilaisia tiedostoformaatteja, joilla kaikilla on omat vahvuudet ja heikkoudet. Lähes kaikkien laitteiden tukemia yleisiä tekstisisältöformaatteja ovat pdf, html ja rtf. Edellisten lisäksi markkinoilla on useita muita tiedostomuotoja, mutta vaikuttaa siltä, että e-kirjojen kohdalla kyseessä on kahden kauppa. Suosituimmiksi tiedostomuo-

doiksi ovat nousseet International Digital Publishing Forumin avoimeen xml-standardiin pohjautuva ePub ja Amazon Kindlen käyttämä azw, joka pohjaa löyhästi aiempaan Mobipocket-formaattiin [Engelen, 2010; Moore, 2009]. Wikipediassa on kirjoitushetkellä listattuna lähes 30 erilaista tiedostomuotoa [Wikipedia, 2011c].

Useat e-lukijat ja lukemisen mahdollistavat multifunktionaaliset laitteet tukevat monia tiedostomuotoja. EPub on kasvattanut suosiotaan vuoden 2011 aikana. Ainut merkittävä lukulaite, josta ePub-tuki vielä tällä hetkellä puuttuu, on Amazonin Kindle. Amazonin e-kirjakauppa on valikoimaltaan eräs maailman suurimmista, mutta kirjat ovat tarjolla ainoastaan azw-tiedostomuodossa ja kirjallisuus pääasiallisesti vain englanninkielistä. Vuonna 2010 Amazonin osuus e-kirjallisuuden myynnistä Yhdysvalloissa oli 58 % (taulukko 2). Kirjojen kustantajat ovat mieltyneet Amazonin oman tiedostomuodon sijaan avoimempaan ePub-formaattiin. Toukokuussa 2011 Amazon julkisti yhdysvaltalaiskustantamoille ja useille käyttäjillekin mieluisan lausunnon, jossa se ilmoitti hyväksyvänsä tulevaisuudessa myös ePub-formaatin digitaalisissa julkaisuissaan [Jones, 2011].



Taulukko 2. Yhdysvaltain e-kirja myynnin markkinaosuudet verkkokaupoittain vuonna 2010 [Keehner et al., 2011]

Amazonin ja ePub-tiedostomuodon liitto olisi hyvä asia, ainakin tavallisen käyttäjän näkökulmasta. Erilaisten tiedostomuotojen viidakko aiheuttaa käyttäjälle päänvaivaa. Olisi tärkeää luoda yhtenäinen standardi tai standardit, jotta käyttäjät voisivat käyttää ostamiaan digitaalisia kirjoja laitteessa kuin laitteessa. Tällä hetkellä erilaiset kopiosuojaukset ja tiedostomuotojen tuen puute hankaloittavat käyttäjän tilannetta ja saattaa olla mahdotonta siirtää aiemmin hankitut kirjat vanhasta lukijasta uudempaan laitteeseen [Engelen, 2010]. Lisäksi formaattien kanssa yhteisymmärrykseen pääsy mahdollistaisi tiedostomuotojen

kehittämisen interaktiivisempaan suuntaan (esim. ePub3) ja tarjoaisi käyttäjälle yhä monipuolisempaa sisältöä kirjoihin.

3. E-kirjojen käytettävyys

ISO-standardi [ISO9241-11, 1998] määrittelee käytettävyyden olevan ”vuorovaikutteisen tuotteen tai järjestelmän käytön tarkoituksenmukaisuutta, tehokkuutta ja miellyttävyyttä määritellyillä käyttäjillä tietyissä käyttötilanteissa”. Jakob Nielsen on laajentanut käytettävyyden määritelmää vielä muistettavuuden, opittavuuden ja virheiden välttämisen ja niistä palautumisen sekä tyytyväisyyden kriteerein [Nielsen, 1994].

E-kirjojen käytettävyydestä puhuessa tulisi ottaa huomioon, että lukukokemuksen käytettävyyteen vaikuttavat monenlaiset seikat. Ne voitaisiin jakaa esimerkiksi seuraavasti:

- lukulaite
- varsinainen e-kirja tiedosto
- lukemiseen tarkoitettu sovellus.

Näiden kolmen kombinaatiosta muodostuu käyttäjän lukukokemus. Lisäksi tulisi muistaa, ettei käyttäjäkokemus rajoitu täysin pelkästään näihin, vaan myös tapa ostaa sisältöä, hallita sitä ja siirtää kirjoja laitteeseen vaikuttaa. Kauppapaikan täytyy siis olla toimiva ja käyttäjän täytyy päästä siellä selaamaan ja tarkastelemaan kaupan olevaa sisältöä, jotta kokonaisuus olisi miellyttävä.

3.1 Nielsenin heuristiikat

Käytettävyyden arvioinnissa käytetään heuristisia sääntöjä, heuristiikkoja, joiden avulla voidaan tehdä heuristisia arviointeja. Nielsenin [1994] määrittelemät 10 käytettävyyden heuristiikkaa ovat tunnetuimmat käytettävyysarvioinnissa käytetyt heuristiset säännöt. Niitä voidaan käyttää monessa tilanteessa, kunhan heuristiikkoja mukautetaan kontekstiin sopivaksi.

Alla on esitetty vapaasti suomennettuna Nielsenin [1994] kymmenen käytettävyyden heuristiikkaa, joita tutkielmani käytettävyysosiossa hyödynnän:

1) Järjestelmän tilan näkyvyys
2) Järjestelmän ja todellisuuden vastaavuus
3) Käyttäjän kontrolli ja vapaus
4) Yhteneväisyys ja standardit
5) Virheiden estäminen
6) Tunnistaminen mieluummin kuin muistaminen
7) Käytön joustavuus ja tehokkuus
8) Esteettinen ja minimalistinen toteutus

9) Virhetilanteiden tunnistaminen, ilmoittaminen ja korjaaminen
10) Opastus ja ohjeistus.

3.2 Kriteereitä e-lukijan suunnitteluun

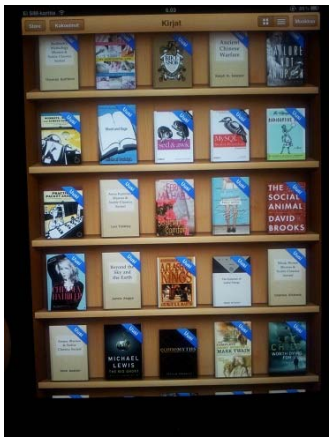
Nielsenin [1994] heuristiikkoihin nojautuen voidaan todeta, että käyttäjät pitävät laitteista, joiden design on yksinkertainen tai jopa pelkistetty (vrt. muut Applen tuotteet). Monet käyttäjät ovat todenneet, etteivät halua lukukokemuksensa häiriintyvän näytön heijastuksien takia. Siksi olisinkin tärkeää, että laitteessa olisi heijastamaton näyttö. Jotta lukukokemus pysyisi miellyttävänä, lukulaite ei saisi myöskään olla liian suuri tai painava [Cleary et al., 2011].

Mikäli näyttö ei ole e-ink-näyttö, tulisi sen kontrastin olla mahdollisimman hyvä. Käyttäjien kannalta merkittävää on myös näytön päivitysnopeus, joka ainakin aiemman sukupolven e-ink-laitteissa jätti reilusti toivomisen varaa. Laitteessa tulisi olla ainoastaan pakolliset näppäimet fyysisinä näppäiminä, mahdolliset muut näppäimet tulisi sijoittaa käytettävien ohjelmien käyttöliittymiin. Fyysisten näppäinten tulisi olla selkeitä (jopa itsestään selviä) ja riittävän kokoisia [Heikkilä et al., 2011].

3.3 Kriteereitä e-kirjan suunnitteluun

Vaikka kysymys on elektronisesta kirjasta, on tärkeää muistaa taustalla oleva metafora. Käyttäjät asettavat elektronisille kirjoille samantyyppisiä odotuksia kuin painetuillekin teoksille ja käyttäytyvät sen mukaan (vrt. sivun kääntämistoiminto kuvassa 3). [Marshall, 2003; Wilson, 2002].

Kuva 3. Sivun kääntämistoiminto

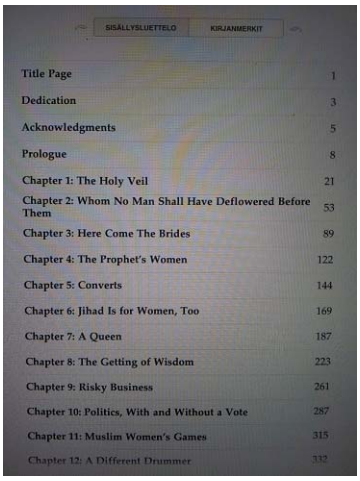


Kuva 4. Kirjahylly

Kirjasta on hyvä olla kansisivu. Se tuo oikean kirjan tuntua ja lisäksi helpottaa suuresti käyttäjän tunnistustehtävää. Mikäli sama kansikuva on näkyvissä ikonina laitteen kirjahyllyssä, niin käyttäjä erottaa kirjan muista kirjahyllyn teoksista, vaikka ei näkisi lukea pienellä näkyvää kirjan nimeä. IBooksin kirjahyllyssä näkyviä kansikuvia on esillä kuvassa 4.

Jos mahdollista tulisi e-kirjassa olla sisällysluettelo, mikäli teoksen tekijä on jaotellut teostaan niin, että sisällysluettelon mukaan sisällyttäminen on ylipäätään mahdollista. Käyttäjät saattavat vierastaa ajatusta, mutta esimerkiksi useimmista Appstoren kautta myytävistä kaunokirjallisista e-kirjoista sellainen löytyy. Tietokirjallisuudessa sisällysluettelon tärkeys korostuu vielä enemmän. Hyödyllisin ratkaisu olisi hyperlinkattu sisällysluettelo, jotta käyttäjä pääsee halutessaan navigoimaan valitsemaansa kohtaan kirjassa suoraan sisällysluettelosta käsin. Sisällysluettelon voi myös katsoa konkretisoivan e-kirjaa käyttäjälle ja auttavan hahmottamaan kirjan rakennetta. Tämä on erityisen tärkeää, koska e-kirjassa käsin kosketeltava tuntuma kirjan sivumäärästä ja rakenteesta ymmärrettävästikin puuttuu [Gibb et al., 2002].

Vaihtoehtoinen navigointi sisällysluettelon kautta on varteenotettava vaihtoehto tai lisä, varsinkin siinä tapauksessa, että varsinaisen lukusovelluksen tarjoama navigointi on puutteellinen [Heikkilä et al., 2011]. Kuvassa 5 on Nine Parts of Desire -kirjan sisällysluettelo.



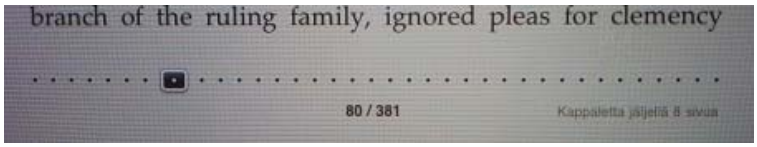
Kuva 5. Sisällysluettelo

Sekä Gibb ja muut [2002] että Heikkilä ja muut [2011] suosittelevat kiinnittämään huomiota tekstin asetteluun kirjassa. Otsikoiden tulisi alkaa uudelta sivulta ja tyhjää tilaa otsikon alla säästelemättä. Ulkoasun tulisi olla esteettisesti tarkkaan harkittu otsikkojen asettelua, välilyöntejä ja sisennyksiä myöden. Tarjolla tulisi olla riittävästi visuaalisia vihjeitä käyttäjälle, jotta kirja ei muistuttaisi epämuodostunutta tekstipötköä.

3.4 Kriteereitä lukusovelluksen suunnitteluun

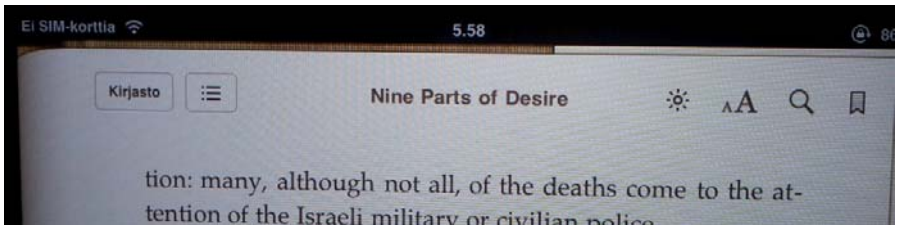
Käyttäjä haluaa tietää, missä vaiheessa kirjaa hän on. Käyttäjälle tulee siis tarjota työkalu etenemisen seuraamiseen, sillä painetun kirjan kohdalla käytetty jäljellä olevan osuuden paksuuden vilkaisu ei tässä tapauksessa toimi. Tätä hae-

taan myös Nielsenin [1994] heuristiikoissa kohdassa ”järjestelmän näkyvyys”. Työkaluja edistymisen seuraamiseksi ovat esimerkiksi sivunumerot sekä mahdollinen navigaatiopalkki, jonka olemassaolo on muutenkin suositeltavaa. Kuvassa 6 on näkyvissä iBooks-sovelluksessa näkyvä alanavigaatiopalkki, josta näkee kerta vilkaisulla, missä kohdassa kirjaa on menossa ja voi halutessaan siirtyä nopeasti eteen tai taaksepäin. Lisäksi iBooks tarjoaa alapalkissa tiedon kappaleessa jäljellä olevien sivun määrästä (kuva 6).



Kuva 6. Alanavigaatio palkki, sivunumerointi ja jäljellä olevat sivut

Nielsenin [1994] järjestelmän tilan näkyvyyden ja käyttäjän kontrolliin ja vapauden heuristiikkoihin pohjautuva toiminto olisi tarjota käyttäjälle sovelluksen tärkeimmät toiminnot jatkuvasti näkyvillä olevana valikkona. Tällöin käyttäjä voi aidosti tuntea hallitsevansa sovelluksen toimintaa ja voivansa tehdä muutoksia nopeasti ja helposti niin tahtoessaan. IBooksissa oleva ylävalikkopalkki on näkyvillä kuvassa 7.



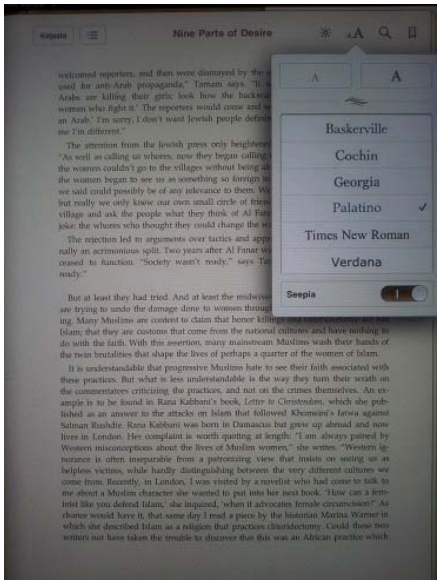
Kuva 7. iBooks-sovelluksen valikkopalkki

Heikkilän ja muiden [2011] mukaan lukusovelluksessa olisi syytä olla tarjolla visuaalinen kirjahylly, johon käyttäjän laitteeseen lataamat e-kirjat ilmestyvät omien kansikuviensa kera. Tämä paitsi tukee mielikuvaa oikeasta kirjasta, myös helpottaa käyttäjää erottamaan haluamansa teoksen useiden mahdollisten joukosta. Kirjahyllyssä tulisi olla useanlaisia lajittelumahdollisuuksia ja hakutoiminto käytön helpottamiseksi.

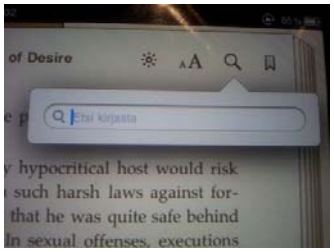
Gibb ja muut [2002] ehdottavat, että käyttäjälle tarjotaan mahdollisuus muokata fontteja ja fonttikokoja laajasti, jotta kaikki erilaiset käyttäjät löytävät omalle näkökyvylle ja silmille sopivat asetus yhdistelmät. Tämä mahdollistaa e-kirjojen lukemisen mm. näkövammaisille. Nielsen [1994] puolestaan suosittelee Sans-Serif -tyyppisten eli pääteviivattomien fonttiratkaisujen käyttöä.

Fonttien tulisi olla tarpeeksi suuria, jotta lukumukavuus säilyy pitkienkin lukuhetkien läpi [Landoni et al., 2004]. Heikkilä ja muut [2011] ehdottavat, että

lukusovelluksessa tulisi olla tarjolla ainakin kuusi erilaista fonttikokoa. Heikkilä ja muut pohjaavat ehdotuksensa tekemäänsä käytettävyystutkimukseen, johon osallistui 18 eri ikäryhmiin kuuluvaa käyttäjää ja jossa testattiin kahdeksaa erilaista e-kirjan lukulaitetta. IBooksin fonttiasetukset on esitetty kuvassa 8.



Kuva 8. iBooksin fonttiasetukset



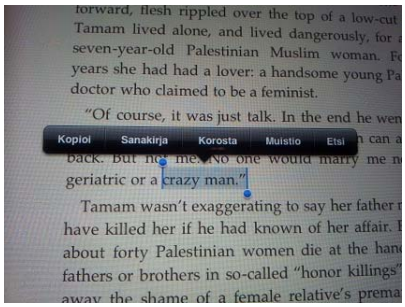
Kuva 9. Hakutoiminto ja kirjanmerkkikuvake

Käyttäjät arvostavat lukusovelluksissa myös muita toimintoja. E-kirjan lukusovellukseen on mahdollista sisällyttää kirjan sisällöstä etsivä hakutoiminto, joten sellaisen poisjättäminen tuntuisi hölmöltä [Gibb et al., 2002]. Kuvassa 9 on näkyvillä iBooksin etsi kirjasta -toiminto.

Reed Collegessa [2010] tehdyssä tutkimuksessa koehenkilöt raportoivat pitäneensä iPadin tarjoamasta mahdollisuudesta tehdä e-kirjojen ja pdf-tiedostojen teksteihin korostuksia ja yliviivauksia ja liittää mukaan omia muistiinpanoja. Samanlaisista tuloksista raportoi myös O'Neill [2009] e-lukijoilla tehdyissä testauksissa.

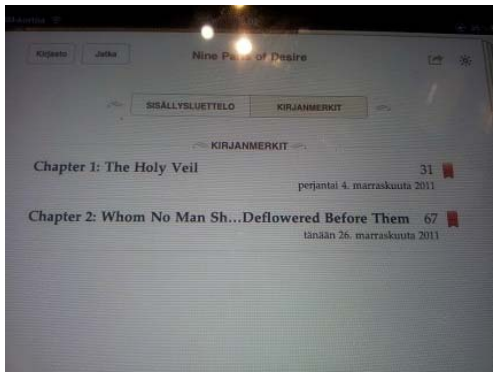
Davis ja muut [2006] raportoivat, että heidän tekemissään tutkimuksissa käyttäjät käyttivät eniten e-kirjojen korostus- ja yliviivaustoimintoa sekä kirjanmerkkaus ja hakutoimintoja.

Applen omassa iBooks-sovelluksessa on yllämainitut toiminnot ja lisäksi se tarjoaa mahdollisuuden tehdä hakuja paitsi kirjan sisällöstä, myös suoraan esimerkiksi Googlen hakukoneella tai sivistyssanakirjasta. Myös tekstin kopioiminen kirjasta onnistuu. Kuvassa 10 on esitelty iBooksissa olevia toimintoja.



Kuva 10. Kopiointi, korostus, hakutoiminnot ja muistiinpanojen tekeminen

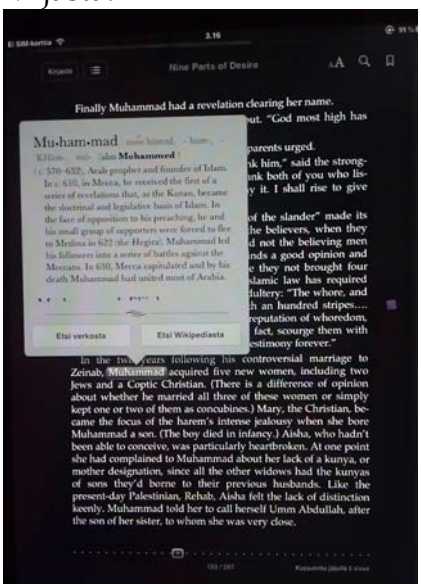
Koska kysymys on kirjasta, on tietenkin luonnollista odottaa myös kirjanmerkkien tekemisen olevan mahdollista (aivan kuten oppilaat Davisin ja muiden [2006] tutkimuksessa olivat toivoneet). Myös Gibb ja muut [2002] suosittelivat kirjanmerkkien tarjoamista. Fyysisen kirjan väliin saa helposti vaikkapa palan paperia ja heikkomoraalisimmat taittavat sivujen kulmia, mutta e-kirjassa nämä lähestymistavat eivät toimi. Koska kirjanmerkki ei voi e-kirjassa olla fyysisesti nähtävillä, on kirjanmerkkien tekemiseen ja havaitsemiseen kiinnitettävä huomiota. Kuvassa 9 ylhäällä oikealla näkyy kirjanmerkki kuvake, jota koskettamalla kyseinen sivu tallentuu iBooksin kirjanmerkkeihin. Kuvassa 11 puolestaan näkyvät iBooks-sovelluksen kirjaan tallennetut kirajnmerkit, joita koskettamalla pääsee jatkamaan lukemista valitusta kohdasta.



Kuva 11. Kirjanmerkit

Kaikkien aiemmin listattujen ominaisuuksien lisäksi e-kirjan lukusovelluksen tulisi Nielsenin [1994] heuristiikkoja seuraten olla mahdollisimman sulavalinjainen ja muotoilultaan ja toiminnaltaan esteettinen. Käytön tulisi olla helppoa ja intuitiivista niin aloittelijalle kuin edistyneellekin käyttäjälle. Lisäksi sovelluksessa tulisi noudattaa jonkinlaista yhtenevää linjaa, jolloin käyttäjä ymmärtää koko ajan, mistä on kysymys ja kuinka edetä. Tällöin käyttäjän oman vapauden ja kontrollin tunteet (joita tavoitellaan) lisääntyvät. Mahdollisuus ostaa kirjoja suoraan laitteesta käsin on helppo tapa parantaa käyttäjän käyttökokemusta lukusovelluksesta.

Taustavalaistun tablet-laitteen käyttäjille olisi hyvä myös antaa mahdollisuus muokata ohjelman näyttöasetuksia. Taustavalaistulta näytöltä lukeminen saattaa rasittaa silmiä vähemmän, mikäli käyttäjä voi valita omaa silmäänsä miellyttävän lukutilan erilaisista vaihtoehdoista. Applen iBooks ohjelman uusimmassa versiossa 1.5 on yölukutila, jota ainakin itse pidän hyödyllisenä. Sen lisäksi jo aiemmin tarjolla oli mahdollisuus vaihtaa valkomusta värimaailma seepian väreihin. Kuvassa 12 on näkyvissä iBooksin yölukutila ja mahdollisuus tutkia valittua sanaa sanakirjasta.



Kuva 12. iBooksin yölukutila ja sanakirjatoiminto

Monikäyttöinen tablet-laite mahdollistaa lukukokemuksen rikastamisen. Tavallista kirjaa lukiessaan käyttäjä saattaa haluta vinkata mieluisasta lukukokemuksestaan lähipiirille ja ystäville, mutta se voi helposti unohtua. Tablet-laitteella näitä vinkkejä ja huomioita on helppo jakaa eteenpäin muutamalla napinpainalluksella sähköpostitse tai sosiaalisen median välityksellä suoraan lukulaitteesta. Tavallisin lukuhetki on illalla ennen nukkumaan menoa. Lukuvalo saattaa häiritä mahdollista kanssanukkujaa ja lukeminen voi jäädä helposti väliin. Taustavalaistulta näytöltä lukiessa valaistusta ei tarvita, joten lukea voi pimeässäkin ketään häiritsemättä.

Sovellukseen lisättävät toiminnot sanakirjasta ja Internetistä hakemiseen tuovat aivan uudenlaisen ulottuvuuden lukemiseen. Käyttäjille, jotka lukevat paljon vieraskielistä tekstiä ja tarvitsevat toisinaan sanakirjaa ja niille, joiden kirjahylly täytyy tieteellisistä teoksista ja katsaus sanakirjaan termistön tarkentamiseksi on silloin tällöin paikallaan, on tablet-laite varsin hyödyllinen lukuväline. Yleisesti ottaen lukusovellusta suunnitellessa tulisi kartoittaa laajasti ne ominaisuudet ja mahdollisuudet, joita lukulaite itsessään tarjoaa ja hyödyntää niitä mahdollisimman monipuolisesti. Vaikka käyttäjät odottavat samantyyppistä lukukokemusta kuin paperisienkin kirjojen kanssa, kaikelta uudelta odo-

tetaan myös jotain enemmän. Virheellinen suunnittelu ei saisi pilata mahdollisuutta tarjota lisäarvoa käyttäjille.

Kiteytys lukusovelluksen suunnittelun kriteereistä:

- Seurannan etenemisen mahdollistaminen (sivunumerot ja seuranta/navigaatiopalkki)
- Tärkeimmät toiminnot jatkuvasti esillä työkalupalkissa (taustavalon määrä ja eri lukutilat, fontti, haku, kirjanmerkkaus ja merkit, kellonaika, pääsy kirjahyllyyn)
- Kirjahyllyn visuaalisuus (hakutoiminto ja lajittelutoiminnot)
- Fonttien ja fonttikokojen muuttamisen mahdollistaminen (esim. Sans-Serif tyyppiset fontit ja ainakin kuusi erilaista fonttikokoa)
- Sivun visuaalinen asettelu (pidetään kiinni sivun määritelmästä)
- Hakutoiminto (sekä kirjasta, että webistä)
- Mahdollisuus tehdä korostuksia, omia muistiinpanoja ja kopioida tekstiä
- Kirjanmerkit, niiden tekeminen ja tarkastelu
- Mahdollisen lisäarvon tarjoaminen käyttäjille (mm. sosiaalisen median integraatioon tukeminen ja muut laitteen tarjoamat mahdollisuudet)
- Kirjojen ostamisen helppous (mahdollisuus ostaa suoraan laitteesta käsin, ilman tarvetta tietokoneelle)
- Lisäksi pidettävä mielessä Nielsenin heuristiikat.

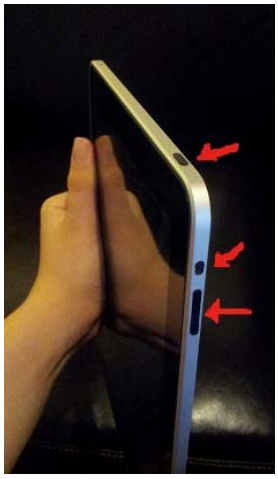
4. Elisa Kirja -sovelluksen käytettävyysarviointi iPad-laitteella

4.1 Arvioinnissa käytetty laite

Käytin arvioinnin tekemisessä Applen ensimmäisen sukupolven iPadia (kuvassa 13). Se ilmestyi markkinoille huhtikuussa 2010. IPadissa on 9,7 tuuman led-taustavalaistu kapasitiivinen kosketusnäyttö ja se painaa mallista riippuen 680 – 730 grammaa. Kaikissa malleissa on Wi-Fi (802.11 abgn) ja tarjolla on myös simkorttipaikalla varustettuja 3g-malleja. Flash-muistia on 16-64 gt. Keskusmuistia on 256 megatavua. Suorittimena on 1Ghz:n Apple A4. [Wikipedia, 2011d].

Kuva 13. Apple iPad [Smartphone Envy, 2010]

Kuva 14. iPadin kotinäppäin



Kuva 15. Virta-, mykistys- ja äänenvoimakkuusnäppäimet

IPad noudattelee muotoilultaan varsin pitkälti Nielsenin [1994] kahdeksatta heuristiikkaa. Suurin osa iPad-laitteesta on pelkkää näyttöä. Laitteessa on ainoastaan neljä fyysistä näppäintä: kotinäppäin (kuvassa 14 alhaalla keskellä), virtanäppäin (kuvassa 15 ylin painike), äänenvoimakkuuden säätö (kuva 15, alin pitkä näppäin) ja mykistys kytkin (kuvassa 15 keskellä). IPadia hallitaan sormin koskettamalla.

IPadissa ei ole valmiina ohjelmaa e-kirjojen lukemiseen, mutta Applen appstoresta on ladattavissa useita erilaisia lukuohjelmia. Tutkielmassa keskityn ainoastaan Applen omaan iBooks-lukusovellukseen ja myöhemmin Elisa Kirja -lukusovellukseen. Molemmilla sovelluksilla on myös omat kirjakauppansa, joista kirjoja voi ostaa ja ladata laitteeseen, tosin Elisa Kirja -palvelua käytettäessä kirjaostokset täytyy tehdä tietokoneella ja vasta sitten ostetun sisällön voi ladata iPadin Elisa Kirja -sovellukseen.

4.2 Elisa Kirja-palvelun käytettävyys

Tehdessäni Elisa-Kirja -sovelluksen käytettävyysarviointia käytin kohdassa 3.4 esitettyjä kriteereitä lukusovelluksen suunnitteluun. Elisa Kirjan iPad-sovellusta

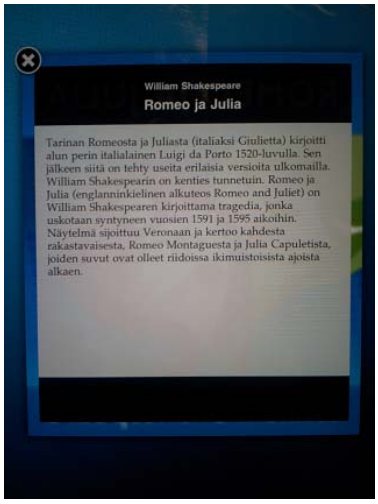
arvioidessani havaitsin, että sovelluksen päävalikossa ei ole lainkaan hakutoimintoa, vaan tarjolla ovat ainoastaan kirjahyllyn kirjojen lajittelu joko kansikuvanäkymään (kuvassa 16) tai listausnäkymään (kuvassa 17). Jos kirjahyllyssä on vain pieni määrä kirjoja, hakutoiminnon puute ei ole häiritsevä. Mikäli teoksia kuitenkin kertyy kirjahyllyyn enemmänkin (enemmän kuin kerralla mahtuu näkyviin), alkaa hakutoiminnon puute varmasti häiritä käyttäjää.

Kuva 16. Elisa Kirjan kirjahyllysivun kansikuvanäkymä

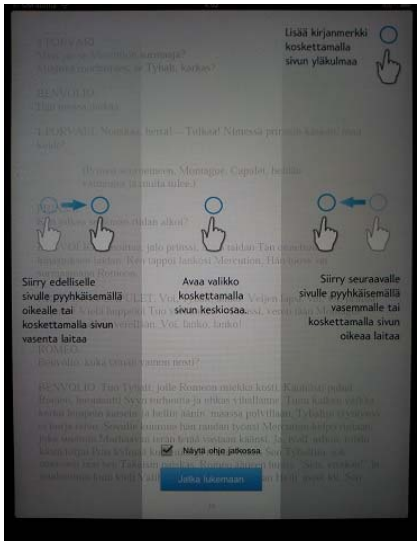
Kuva 17. Elisa Kirjan kirjahyllysivun listausnäkymä

Sovelluksen kirjahyllyn kansikuvanäkymä-sivu (kuvassa 16) muistuttaa iBooksin vastaavaa. Kansikuvista on hyvin erotettavissa teoksen nimi, mutta heikkonäköisemmältä voi jäädä teoksen tekijän nimi lukematta, koska se on esitetty huomattavasti pienemmällä fontilla. Listausnäkymässä (kuvassa 17) puolestaan sekä teoksen että tekijän nimi ovat mielestäni esitetty riittävän suurella fontilla. Kuvassa 17 rivien oikeassa reunassa näkyvä i-painike avaa kirjan takakannen käyttäjän luettavaksi. Pidin tätä erittäin hyvänä. Tosin kansikuvasivulle olisin kaivannut jotain visuaalista vihjettä takakannen näkyviin saamiseksi. Takakansi on näkyvillä kuvassa 18. Käyttäjältä saattaa huonoimmassa

tapauksessa jäädä koko toiminto huomaamatta, mikäli ei tule painaneeksi kirjan kuvaketta pitkään pohjassa. Sovelluksen kirjahyllyn sisältö on mahdollista lajitella lukuhetken, kirjailijan ja teoksen nimen mukaan ja mielestäni tämä lajittelu onkin riittävä.



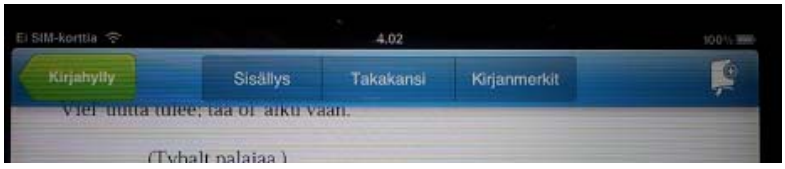
Kuva 18. Elisa Kirja takakansi näkymä



Kuva 19. Elisa Kirjan ohjesivu

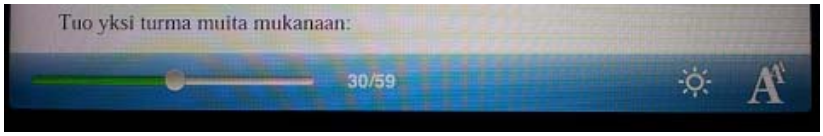
Käyttäjän valitessa kirjahyllystä jonkin teoksen luettavakseen aukeaa kuvan 19 kaltainen ohjenäkymä. Tämä on erityisen hyödyllinen toiminto, varsinkin niille käyttäjille, jotka eivät ole aiemmin vastaavanlaista lukulaitetta tai sovellusta käyttäneet. Edistyneemmät käyttäjät eivät ehkä ohjetta kuitenkaan kaipaa, joten mahdollisuus estää sivun aukeaminen on hyvä ja tarpeellinen ominaisuus. Itse en kuitenkaan löytänyt sovelluksesta mahdollisuutta kytkeä ohjetta takaisin näkyviin, kun sen oli kertaalleen ottanut pois näkyvistä. Jos laite on vain yhden henkilön käytössä, tämä ei muodostu ongelmaksi. Jos käyttäjiä on enemmän kuin yksi, ohjeesta pääsee nauttimaan vain ensimmäinen käyttäjä, mikäli se kytketään pois päältä ensimmäisen näyttökerran jälkeen.

Kuvissa 20 ja 21 nähdään sovelluksen ylä- ja alapalkki, joissa sovelluksen tärkeimmät toiminnot ovat esillä. Palkit voi halutessaan pitää jatkuvasti näkyvissä tai piilottaa. Piilottaminen ja näkyviin saaminen onnistuu sivun keskiosaa koskettamalla. Yläpalkki peittää näkyessään osan ylimmästä (kuvassa 20) tekstirivistä ja olisi hyvä jos tämän pienen, mutta häiritsevän ongelman saisi korjattua.



Kuva 20. Elisa Kirjan yläpalkki

Sovelluksen yläpalkkiin on toteutettu navigaatio kirjahyllyn, sisällyksen, takakannen ja kirjanmerkkien välille. Lisäksi oikealla ylhäällä on kirjanmerkin tekemiseen tarkoitettu painike. Pidän sovelluksen navigointiratkaisua selkeänä ja hyvänä. Asia, joka jäi mietityttämään, oli takakansi näkymän tarpeellisuus yläpalkissa.



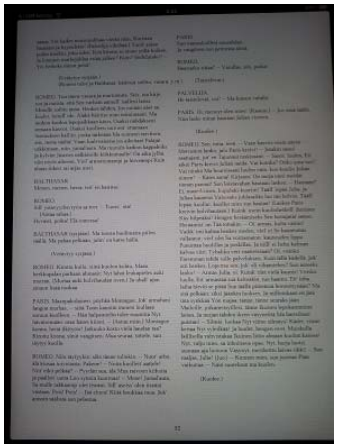
Kuva 21. Elisa Kirjan alapalkki

Alapalkissa (kuva 21) tarjotaan mahdollisuus navigoida kirjassa nopeasti eteen- ja taaksepäin liu'utettavan palkin avulla. Navigaatiopalkin vieressä on näkyvillä ko. sivun numero ja lisäksi koko kirjan sivumäärä. On hyvä, että käyttäjä näkee koko ajan, missä kohtaa kirjaa on menossa. Jos haluaisi tehdä muutoksia, niin halutessaan käyttäjälle voisi ilmoittaa ko. luvussa jäljellä olevien sivujen määrän. Tämän puuttuminen ei kuitenkaan ole mielestäni suuri puute. Alapalkin oikeasta laidasta löytyvät kirkkaus- ja fonttikokosäädöt (kuva 22). Fonttikokoja on tarjolla viisi erilaista ja jäinkin kaipaamaan niitä enemmän. Suurena puutteena pitäisin sitä, ettei käyttäjälle tarjota lainkaan mahdollisuutta vaihtaa fonttia. Käytössä oleva fontti ei ole pääteviivaton kuten esimerkiksi Nielsen [1994] nimenomaan suosittelee. Pääteviivattominen eli Sans-Serif-kirjasintyyppien lukeminen on selkeämpää ja helpompaa kuin pääteviivallisten Serif-kirjasintyyppien. Suosittelisin, että sovellukseen lisättäisiin mahdollisuus vaihtaa fontteja ja perusfontiksi asetettaisiin jokin pääteviivaton fontti. Mikäli fonttivaihtoehtoja ei haluta antaa yhtä enempää, ehdottaisin että ainakin perusfontti vaihdettaisiin pääteviivattomaan lukumukavuuden lisäämiseksi.

Kuva 22. Elisa Kirjan fonttiasetukset

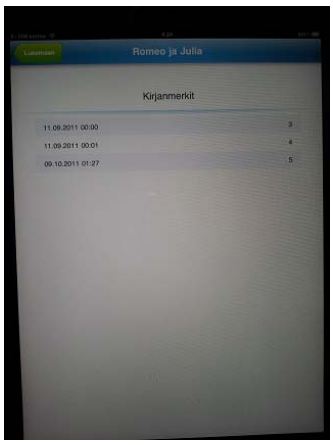
Koin ongelmalliseksi myös sen, että sovelluksen pienimmällä fontilla lukiessa sivu jakautuu kahteen palstaan (kuvassa 23). Tämä ei missään tapauksessa ole esteettisesti hyvä ratkaisu. Sivun perinteisen rakenteen rikkomisen lisäksi siinä menetetään myös sivunumeroiden tuoma hyöty, sillä fontin pienimmällä

asetuksella sivunumerot pomppaavat kaksi kerrallaan eteenpäin. Myös suurinta fonttia käyttäessä törmää sivunumerojen kannalta epäloogiseen tilanteeseen, jossa sivua kääntämälläkin voidaan olla esimerkiksi kolmen eri sivun aikana samalla sivunumerolla. Nämä kaksi epäkohtaa korjaisin ehdottomasti.



Kuva 23. Elisa Kirjan fonttikoko pienimmillään

Sovellus tarjoaa mahdollisuuden tehdä kirjanmerkkejä yläpalkin oikeasta laidasta (kuvassa 20) kirjanmerkki-kuvakkeesta. Tehtyjä kirjanmerkkejä (kuvassa 24) pääsee tarkastelemaan yläpalkin kirjanmerkit painiketta painamalla. Kirjanmerkki-sivulla näkee tehdyt kirjanmerkit etenemisjärjestyksessä. Lisäksi näkyy kirjanmerkin tallentamispäivä- ja aika, mutta ei esimerkiksi sen luvun nimeä, jossa ollaan menossa kunkin kirjanmerkin kohdalla. Luvun nimen puute ei ole varsinainen käytettävyysongelma, mutta sen lisääminen olisi käyttäjän kannalta mieluisa muutos.



Kuva 24. Elisa Kirjan kirjanmerkkisivu

Kuva 25. Elisa Kirjan sisällysluettelosivu

Jäin kaipaamaan mahdollisuutta muuttaa lukutilan väritystä. Tarjolla voisi olla mahdollisesti seepiaväritys ja iBooksin tapaan yölukutila, jossa tausta on musta ja fontti vaalea. Erilaiset lukutilat voisivat tuoda lisäarvoa ja käyttömukavuutta niille käyttäjille, joiden silmät rasittuvat mustan tekstin lukemisesta

valkoiselta pohjalta taustavalaistulla näytöllä. Lisäksi yölukutila tarjoaa käyttäjälle mahdollisuuden lukea samassa tilassa nukkuvia häiritsemättä.

Sovelluksen sisällysluettelonäkymä (kuvassa 25) on tavanomaisen näköinen ja luettavasta kirjasta riippuen tarjolla on hyperlinkkaus, eli käyttäjä pääsee valittuun kohtaan kirjassa suoraan sisällysluettelosta haluamaansa lukua painamalla.

Merkittävänä puutteina näen sen, että Elisa Kirja -sovellus ei anna käyttäjälle minkäänlaista mahdollisuutta hakea valittua tai kirjoitettua tekstiä suoraan kirjan sisällöstä tai sanakirjasta tai Internetistä. Tarjolla ei myöskään ole mahdollisuutta tekstin kopioimiseen tai omien muistiinpanojen ja korostuksien tekemiseen. Laitealusta tarjoaa mahdollisuuden edellä mainittujen ominaisuuksien toteuttamiseen, joten suosittelisin niiden lisäämistä palveluun. Tällaiset ominaisuudet tuovat käyttäjille suurtakin lisäarvoa, eritoten jos Elisa Kirja -palveluun halutaan joskus tuoda myyntiin muutakin kuin kaunokirjallisuutta. Tietokirjallisuutta ja oppimateriaaleja luettaessa edellä mainitut toiminnot ovat erittäin tarpeen.

Nostaisin esiin sivunkääntöanimaation, jonka jouhevuus ja esteettisyys jättää hieman toivomisen varaa. Pienellä hiomisella animaation saisi aidomman näköiseksi ja silmää miellyttävämmäksi. Visuaalinen ilme paranisi suuresti, jos sivun animaatio muotoutuisi sen mukaisesti, mistä kulmasta sivun kääntämisen aloittaa. Viimeisenä kiinnittäisin huomiota Elisa Kirjan iPad-sovelluksen kirjojen hankkimistapaan. Kirjojen ostaminen ei onnistu suoraan sovelluksesta, vaan vaatii pääsyn Elisa Kirjan Internetsivustolle. Missään kohtaa sovellusta ei mainita, että käyttäjä voi tehdä kirjaostoksensa vaikkapa iPadin selaimen kautta ja sitten vain ladata ostokset sovellukseen. Moni käyttäjä jääkin varmasti siihen käsitykseen, että kirjojen saamiseksi laitteeseen tarvitaan tietokonetta ja näin ollen kirjaostokset saattavat jäädä reissussa ollessa tekemättä. Parasta tietysti olisi, jos kirjat voisi ostaa suoraan sovelluksesta itsestään.

Elisa Kirja -sovelluksen hyvät puolet

- Takakansi (kaipaisi ehkä vihjeen kirjahylly näkymässä ja on toisaalta hieman turha yläpalkissa)
- Hyvä ohje (tosin ongelma on, ettei ohjetta saa uudelleen näkyville, jos sen ottaa pois näkyvistä)
- Sisällysluettelon hyperlinkkaus
- Päätoiminnot näkyvillä yläpalkissa
- Käyttäjä pystyy seuraamaan etenemistään hyvin alapalkin kautta ja tarjolla on nopea navigointi

Elisa Kirja -sovelluksen huonot puolet

- Päävalikossa ei lainkaan hakutoimintoa
- Yläpalkki peittää osittain ylimmän tekstirivin
- Fonttikokojen vähyys ja kiinteä fontti jota ei pääse muuttamaan (Serif tyyppinen fontti huono)
- Ei seepia tai yölukutiloja
- Pienimmän fontin käyttö jakaa sivun kahteen palstaan
- Suurilla fonteilla sivunumerointi sekaisin (monen sivun aikana ollaan samalla sivunumerolla)
- Kirjanmerkkien nimeäminen (nyt nimettynä ainoastaan ajankohdan mukaan)
- Ei mahdollisuutta kopioida tai yliviivata tekstiä
- Ei mahdollisuutta tehdä omia muistiinpanoja
- Ei mahdollisuutta hakea kirjan sisällöstä tai Internetistä
- Animaation sujuvuus
- Kirjojen ostaminen ei onnistu suoraan iPad-sovelluksesta

5. Yhteenveto

E-kirja on uusi tulokas Suomessa, mutta se on onnistunut herättämään kiinnostusta niin kuluttajissa kuin palveluntarjoajissakin. Osoituksena tästä ovat mm. Elisa Kirja -palvelu ja Suomalaisen kirjakaupan tarjoama lukusovellus, joista kirjoja voi ostaa tablet-laitteeseen. E-kirjamarkkinoille on luvassa kasvua niin maailmanmarkkinoilla kuin Suomessakin, mutta vielä tällä hetkellä tarjonta ei pysty Suomessa kilpailemaan täysipainoisesti painetun kirjallisuuden kanssa.

Aiemmin e-kirjojen lukemiseen tarvittiin erillistä lukulaitetta ja harva halusi sellaisen hankkia. Monikäyttöisten laitteiden yleistyessä tilanne muuttuu jatkuvasti. Esimerkiksi tablet-laitteiden huima yleistyminen viimeisen parin vuoden aikana voisi antaa viitteitä siitä, että elektronisten kirjojen markkinat ovat kovassa kasvussa. Kun ihmiset tutustuvat tarkemmin omistamiinsa laitteisiin ja niiden tarjoamiin mahdollisuuksiin alkaa e-kirjallisuus kiinnostaa yhä useampia. Oma lukunsa on oppimateriaalien sähköiseen muotoon siirtäminen. Suunnitelmat siitä on tehty aikaa sitten, mutta toteutus on ollut hidasta. Hyviin tuloksiin on päästy kuitenkin esimerkiksi Yhdysvalloissa, jossa pelkästään New Yorkin kaupungin kouluihin on hankittu iPad-laitteita yli miljoonan euron edestä. Suomenkin korkeakoulumaailmassa kurssien luento-oppimateriaalit ovat pääosin tarjolla sähköisessä muodossa.

E-kirjojen yleistyessä tulevaisuudessa, tulee käytettävyyden merkitys korostumaan. Kun kilpailijoita on paljon, käyttäjät valitsevat palvelun, jonka käytön kokevat helpoksi ja miellyttäväksi.

Viiteluettelo

[Apple, 2011] Apple Computer Inc, iCloud. Saatavilla http://www.apple.com/fi/icloud/. Tarkistettu 13. joulukuuta, 2011.

[Cleary et al., 2011] Colleen E. Cleary, Ann Huthwaite, Alex McClintock, Brendan Sinnamon and Peter Sondergeld, Ebook readers: separating the hyper from the reality. In: P*roc. of 2011 ALIA Information Online Conference & Exhibition*, Sydney Convention & Exhibition Centre.

[Davis et al., 2006] Darcy Davis, Herbert Dershem and Ryan McFall, Experieces using a collaborative electronic textbook: bringing the "guide on the side" home with you. In: *Proc. of the 37th SIGCSE '06 Technical Syposium on Computer Science Education*, ACM USA, 2006, 339-343.

[E Ink, 2010] E Ink Corporation, Feature technology ink. Available as http://www.eink.com/images/feature_technology_ink.gif. Checked October 28, 2011.

[Engelen, 2010] Jan Engelen, E-books: finally there? Available as http://elpub.scix.net/data/works/att/134_elpub2010.content.pdf. Checked November 15, 2011.

[Gibb et al., 2002] Forbes Gibb, Monica Landoni and Ruth Wilson, Guidelines for designing electronic books. In : *Proc. of the Sixth European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2002)*, Berlin, 2002. 47-60.

[Goleman and Norris, 2010] Daniel Goleman and Gregory Norris., How Green Is My iPad? Available as http://www.nytimes.com/interactive/2010/04/04/opinion/04opchart.html. Checked December 10, 2011.

[Groner et al., 2010] Rudolf Groner, Eva Siegenthaler and Pascal Wurtz, Improving the usability of e-book readers. *Journal of Usability Studies* **6** (Nov. 2010), 25-38.

[Heikkilä et al., 2011] Harri Heikkilä, Kajsa Hytönen, Merja Helle, Jan Kallenbach, Karri Kallinen and Niklas Rajava. Ereading, media use, experience and adoption. Next media – a Tivit programme. Report D1.1.4.2, March 2011. Also available as: http://virtual.vtt.fi/virtual/nextmedia/Deliverables-2010/D1.1.4.2_D1.1.4.3_D1.1.4.4%20eReading_Media_Use,%20Experience%20and%20Adoption.pdf. Checked November 5, 2011.

[Henke, 2002] Harold Henke, Survey on electronic book features. Open ebook forum survey, March 2002. Also Available as http://www.immagic.com/eLibrary/ARCHIVES/GENERAL/IDPF_US/I020320H.pdf. Checked November 15, 2011.

[IDC, 2011] International Data Corporation – Press Release. Media tablet and ereader markets beat second quarter targets, forecast increased for 2011, according to IDC. September 2011. Available as http://www.idc.com/ getdoc.jsp?containerId=prUS23034011. Checked October 14, 2011.

[ISO9241-11, 1998] Int. Organization for Standardization, Ergonomic requirements for office work with visual display terminals (VDTs), Part 11: Guidance on usability. Switzerland, Genève, 1998. Also available as http://www.it.uu.se/edu/course/homepage/acsd/vt09/ISO9241part11.pdf. Checked December 13, 2011.

[Jamali et al., 2008] Hamid R. Jamali, David Nicholas and Ian Rowlands, Scholary e-books: the views of 16,000 academics results from the JISC national e-book observatory. In: *Aslib Proceedings: New Information Perspectives* **61**, 1 (2009), 33-47.

[Jones, 2011] Philip Jones, Amazon to accept epub files. Available as http://www.thebookseller.com/news/amazon-accept-epub-files.html. Checked November 5, 2011.

[Keehner et al., 2011] Jonathan Keehner, Jeffrey McCracken and Matthew Townsend. Barnes and Noble said to be likely to end search without buyer. Available at http://www.bloomberg.com/news/2011-03-22/barnes-noble-is-said-to-be-likely-to-end-search-for-buyer-without-a-sale.html. Checked November 30, 2011.

[Landoni and Wilson, 2002] Monica Landoni and Ruth Wilson, EBONI Electronic Textbook Design Guidelines. March 2002 Available at: http://ebooks.strath.ac.uk/eboni/guidelines/Guidelines.pdf. Checked November 15, 2011.

[Landoni et al., 2004] Chrysanti Malama, Monica Landoni and Ruth Wilson, Fiction electronic books: a usability study. In: *8th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2004),* United Kingdom.

[Lebert, 2009] Marie Lebert, A Short History of eBooks. Available as http://www.etudes-francaises.net/dossiers/ebookEN.pdf. Checked November 16, 2011.

[Li, 2008] Yu Li, Dedicated E-reading Devices: the State of the Art and the Challenges. Scroll Essays on the design of electronic text by graduate students ant the iSchool, University of Toronto, 2008. Also available as

http://fdt.library.utoronto.ca/index.php/fdt/article/view/4908/1770. Checked December 12, 2011.

[Marmarelli and Ringle, 2010] Trina Marmarelli and Martin Ringle, The reed college ipad study. Available as http://www.reed.edu/cis/about/ipad_pilot/Reed_ipad_report.pdf. Checked October 10, 2011.

[Marshall, 2003] Catherine C. Marshall, Reading and interactivity in the digital library: creating an experience that transcends paper. In: *Proc. of the CLIR Kanazawa Institute of Technology Roundtable*, (2005), 3-4

[Moore, 2009] Lisa Moore, At your leisure: assessing ebook reader functionality and interactivity. Project report submitted in partial fulfillment of the requirements for the degree of Master of Science, University College, London, 2009. Available as http://www.uclic.ucl.ac.uk/distinction-projects/2009-Moore.pdf. Checked November 20, 2011.

[My eBook Design, 2011] eBook Design Advantages. Available at http://www.myebookdesign.com/E-Book-Design-Advantages.aspx. Checked November 2, 2011.

[Nielsen, 1994] Jakob Nielsen, *Usability Engineering*. Morgan Kaufman Publishers, 1994.

[O'Neill, 2009] Laura C. O'Neill, A usability study of e-book platforms. A Master's paper, University of North Carolina, 2009. Available as http://ils.unc.edu/MSpapers/3512.pdf. Checked December 4, 2011.

[Oxford Dictionaries, 2011] Oxford Dictionaries. November 2011. Available as http://oxforddictionaries.com/definition/e-book?q=e-book. Checked November 5, 2011.

[Peltoniemi, 2011] Peltoniemi, Jari. Digikirja kiinnosti messukävijöitä. Saatavilla http://www.prosessori.fi/uutiset/uutinen2.asp?id=58564. Tarkistettu 3.11. 2011.

[Pepitone, 2011] Pepitone Julianne. E-book sales top paperbacks for first time. Available as http://money.cnn.com/2011/04/15/technology/ebooks_beat_paperbacks/index.htm. Checked 12.11.2011.

[Pratt, 2010] Keryn Pratt, Netbook, ereader, or ipad? – that is the question. *Computers in New Zealand Schools: Learning, Teaching, Technology*, **22**, 2010.

[Savov, 2011] Vlad Savov, Hanvon brings world's first color e ink reader to CES we go hands-on. Available as http://www.engadget.com/2011/01/09/ hanvon-brings-e920-worlds-first-color-e-ink-reader-to-ces-we/. Checked October 28, 2011.

[Schcolnik, 2001] Miriam Schcolnik, A study of reading with dedicated e-readers. Doctoral thesis, Graduate School of Computer and Information Sciences Nova Southeastern University. 2001. Available as

http://www.planetebook.com/downloads/schcolnik.pdf. Checked November 15, 2011.

[Smartphone Envy, 2010] Apple iPad. Available as http://www.smartphoneenvy.com/wp-content/uploads/2010/10/apple-ipad.jpg. Checked November 20, 2011.

[Topten reviews, 2011] 2012 Best ebook reader comparisons and reviews. Available as http://ebook-reader-review.toptenreviews.com/. Checked October 14, 2011.

[Wikipedia, 2011a] Wikipedia, Gutenberg-projekti. Saatavilla http://fi.wikipedia.org/wiki/Gutenberg-projekti.. Tarkistettu 15.11.2011.

[Wikipedia, 2011b] Wikipedia, E-Book. Available as http://en.wikipedia.org/wiki/E-book. Checked at November 2, 2011.

[Wikipedia, 2011c] Wikipedia, Comparison of e-book formats. Available as http://en.wikipedia.org/wiki/Comparison_of_e-book_formats. Checked October 27, 2011.

[Wikipedia, 2011d] Wikipedia, iPad. Available as http://en.wikipedia.org/wiki/IPad. Checked September 29, 2011.

[Wilson, 2002] Ruth Wilson, The "look and feel" of an e-book: considerations in interface design. In*: Proc. of the 2002 ACM symposium on Applied Computing*, (2002).

[Woo, 2011] Stu Woo, E-book lending takes off. Available as http://online.wsj.com/article/SB10001424052748703726904576192923709743108.html. Checked November 22, 2011.

Reducing the attack surface of an operating system

Ville Valkonen

Abstract:

Certain security choices done on the operating system level can mitigate harm caused by malicious activities. The main focus in the thesis is on reducing the attack surface of open source operating systems.

Keywords: Software security, system security
CR-classes: D.4.5, D.4.6

1 Introduction

This thesis examines different proactive security methods that can be integrated into operating systems and therefore reduce the damage made by various malicious activities. Some of the methods are easier to adopt in general programming guidelines while others are more dependent on the environment. The main focus of this thesis is on open source operating systems.

The difference between the high secure systems and non-secure system can be described in the following way: non-secure systems concentrate on implementing certain functionality while secure system's main goal is to achieve functionality in a safe manner. Designing a secure system is an arduous process and difficult to get correct. Basically, a system should *fail safely*, should run the *least privilege* and should not be too complicated to understand (*KISS*). These topics are discussed in greater detail below. When studying and implementing secure systems, one should member what Erik Poll [12] has stated: "The attacker only has to get lucky once, the defender has to get it right all the time".

2 Preliminares

This chapter focuses on different technologies and methods that are widely adopted by many known security environments, programs and security fo-

cused operating systems. Hence, systems that lack of exhaustive testing process or are in a prototype state are excluded from this thesis. This chapter also presents shortages that current approaches might have. This thesis tries to give a general understanding of operating system vulnerabilities that can be pursued − but does not try to cover all possible fields or methods. There are fields regarding concurrent processing, networking and input/output processing that are omitted in this thesis, though some of them might be mentioned briefly.

We start by defining what is a software flaw and what proactive functions can be done to prevent or reduce the damage. The software flaw is an unexpected operation of a program. It can expose system to a great danger, leak confidential information or crash the whole system. On some cases, especially in network related environments, the software flaw can pose huge number of computers to a danger. If it is possible to gain more information from the system than it is designed to offer, or when the system is set up to work in a way it was not intended, it can be seen as a vulnerability.

There are several different types of vulnerabilities. One can measure vulnerabilities according to their severity, e.g., by measuring their level of criticality. Security companies rank vulnerabilities in a different way but certain parts of the rankings are the same. One of the unarguable ranking category is the type of domain, *a local exploit* and *a remote exploit*. The local exploit demands that attacker must have a local access into the system. The local access can be physical or non physical (a user account in the machine). Instead, the remote exploit does not have this demand and it can therefore be seen more dangerous. Remote exploits are considered more risky since there is no need to have the account in the machine. Hence, vulnerabilities can be exploited via Internet.

Communication in the Internet is based on packet data. Thus, a client has to negotiate connection to a server (in normal cases). Negotiating the connection defines a common language for both parties, this is better known as *a communication protocol*. Reporting the *user agent* information is part of the communication process. Without including the exceptions user agent information usually includes a program version. This information defines whether the server and the client can communicate together, i.e., the client has apt version of a program. By examining the communication information,

a malicious user can gain enough information to pursue the functional attack against the service.

Evans and Larochelle [3, pp. 8-9] demonstrated the use of the *splint* [17] tool against *wu-ftpd* [22] daemon. At the time their paper was released wu-ftpd was a widely used ftp daemon [1]. Splint is *a statistical code analyzer* that detects certain types of weaknesses from the program by analyzing its source code. Detecting a basic buffer overflow and misuse of functions is a part of splint's reportoire. By examining the tool result Evans and Larochelle found bugs that were not found before, as well as bugs that were already known by the vendor. Although they used statistical code analyzer against the source code, it is possible to practice exploit scenario before fulfilling it. Furthermore, as the analyzed program was interacting with other computers, it is possible to use studied information to arbitrary exploit daemons in the Internet.

In addition, it is possible to use code annotations where the especially crafted syntax is inserted into the code. This functionality needs support from the compiler. Annotations makes it possible to add more fine-grained checks into the code and get better understanding of the program. Anybody can review open source software and review the code, use tools like splint to analyze the code and to generate an exploit. This can be seen as a bad and a good thing. Bad, as people can read the code and spot flaws more easily. Good, because code gets more reviews and bugs get fixed by anyone that has certain understanding. This certainly needs interaction with the community. Unless explicitly mentioned, all examples in this thesis are related to a Unix based operating system.

2.1 Memory

Different kinds of attack methods that can be pursued via memory are reviewed in this section. Moreover, this helps to gain better understanding regarding the protection methods.

Since it is easier to understand the process memory layout by having a concrete real life example (see Figure 2.1 and Code fragment 2.1), we use the same code that Wilander and Kamkar [21, p. 3] used to describe the process.

[1] In Unix, daemon is a background process.

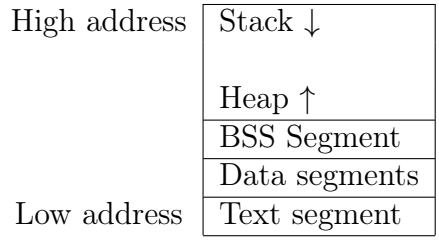


Figure 2.1: Memory layout of the Unix process. [21, p. 2]

By examining the source code in Code fragment 2.1 we can conclude that arguments, constants and variables are stored in the different places in the stack (see Figure 2.2). For instance, statically declared integer variables are stored in the BSS stack (Figure 2.1). Machine code is the deviant in the memory layout that cannot be observed from the source code example. It is stored into the text segment.

```
1 static int GLOBAL_CONST = 1; // Data segment
2 static int global_var; // BSS segment
3
4 // argc & argv on stack, local
5 int main(argc **argv[]) {
6     int local_dynamic_var; // Stack
7     static int local_static_var; // BSS segment
8     int *buf_ptr=(int *)malloc(32); // Heap
9 }
```

Code fragment 2.1: Memory layout elements in code.

Every time a function call occurs, the stack grows by one (will be the top element in this implementation). Each of these calls includes local variables, old base pointer, return address and arguments.

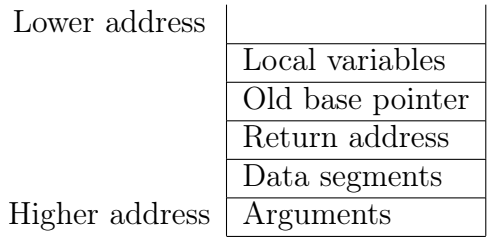


Figure 2.2: A stack frame. [21, p. 3]

Various attack methods concentrate on altering the stack frame in such a way that the attacker can execute arbitrary code and potentially gain higher privileges.

2.2 Buffer overflow

Over decades buffer overflow attacks have been populating the software security flaw top-lists [2]. Buffer overflow occurs when a process writes to the memory address out of the allocated area. Buffer overflow enables code injection exploitation through the current process. This is known as an arbitrary code execution. A demanded attacker can gain himself or herself the same privileges that the process has.

```
1 char mymsg[2] = "hi";
2 strcpy(mymsg, "hello world!");
3 printf("\%s\textbackslash{n}", mymsg);
```

Code fragment 2.2: Simple buffer overflow example in C language (only the relevant parts) is shown in the code fragment.

In Code fragment 2.2, line 1 contains the first flaw: length of the *mymsg* string is 2 but strings should be terminated by the *NUL-terminating character* \0. NUL-termination is crucial since many functions in C depend on that. Without termination character function does not know when to stop. It continues until some memory address includes one.

In line 2, *strcpy* function is used for copying a string *hello world!* into *mymsg*. Since the size of the *hi* is smaller than the copied string *hello world!* buffer overflow occurs. This makes the program indeterministic.

Although one certain type of overflow was examined in Code fragment 2.2, several variations of buffer overflow exist. Theo de Raadt [16] lists the following variations: stack overflows, data segment overflows, GOT/PLT overwrite, jumping to data that attacker can execute, four byte modification possibilities and four byte read possibilities.

2.3 Access control methods

Access control methods limit the access of a certain resource. They can be also used to limit visibility. Access control must not be mixed with authen-

tication. Access control checks whether a user has permission to complete certain activity, in contrast to authentication which identifies user [23]. Several access control methods exist for different purposes. Some offer greater granularity and are therefore more flexible. This gives better control for administrators.

Discretionary access control (DAC) is the simplest access control mechanism that is discussed in this thesis. It has gained popularity among Unix operating systems, since it has fairly clear structure that follows the KISS design principle. Nowadays many modern operating systems have moved towards more fine-grained access control, such as the access control list (ACL) and the mandatory access control list (MAC).

The DAC has only two main features, ownerships and delegations. A user who created an object is also an owner of the object. Owners have all the rights for the object and owner can delegate certain rights to the users.

Authorization states can be expressed as an access control matrix which includes two properties: subjects and objects. Hence, columns present objects and rows present subjects.

	File 1	**File 2**	**Program**
Alice	own read write		
Bob	read	read write	execute
Charlie		read	own read write

Figure 2.3: DAC access matrix.

Figure 2.3 depicts access control matrix in Unix system. Despite the fact that DAC is pretty flexible for regular use, it does not control how information is forwarded. Therefore, a trojan horse attacks are completely viable. Trivellato [23] gives an example where Alice grants Bob with reading permission but does not want Charlie to be able to read the information (see Figure 2.3). Bob can leak information to Charlie without even knowing it. Attack is achievable by using the trojan horse technique which was shown in greater detail by Trivellato (see [24, p. 11]).

ACL expands DAC so that one file can have different access control rules. With ACL it is possible to attach rules like *Allow Bob read and write the File1, Charlie to read and Alice to write* [5].

3 Coping methods and discussion

3.1 Design principles

Several design principles that are known to improve error handling, increase the safety and the controlling abilities should be used in security oriented programming. Obeying these principles does not guarantee flawless programs but helps to cope with the problem. These principles should be usually performed at the beginning of the design process, since it is easier to design functionality around them. It is viable to adopt the methods later on but usually it means vast changes into the code.

3.1.1 KISS

KISS is an acronym for Keep It Simple, Stupid! [4]. Principle encourages simplicity over perfection. According to the principle, more lines of code implies more obscure structures and therefore makes it harder to understand. Updating the code comes harder and implementing a new functionality can be challenging. In general, more lines of code increases bug count and poses more (security) flaws. Moreover, program becomes less deterministic and ambiguous at many levels. Original Unix was designed to adhere the KISS principle, therefore one tool is designed to accomplish one operation.

3.1.2 Whitelists

Whitelist is opposite of blacklist. Instead of denying forbidden functionality whitelist allows certain functionality or operation. If the capability needs are not met it is denied by default.

An excellent usage of whitelisting is presented in the program *TCP wrappers* [18]. TCP wrapper is used for limiting hosts to access services. In addition to whitelisting, it allows blacklisting as well.

3.1.3 Least privilege

When running a process, it should not use higher privileges than is needed. Therefore, if possible, process should drop all extra privileges after the required high privilege job has been completed.

Example 3.1. HTTP daemon needs to bind to port 80 (HTTP traffic) in order to be able to serve web pages for web browsers. Ports that are equal to or lower than 1023 are called as *low ports*. In order to bind socket to these ports, *super user* privilege [2] is required to accomplish the task. After completion the port binding daemon should drop all extra privileges that are no longer needed.

However, as Provos et al. [14, p. 2] state, this does not guarantee safeness. Certain type of flaw in a daemon process can still leak higher privileges to a malicious attacker. As a countermeasure, Provos et al. propose *privilege separation* as an addition to the least privilege. For more details about privilege separation on below.

3.1.4 Fail securely

When malfunction happens it must be taken care an appropriate manner. If one level of security fails there should be another level to mitigate malfunctioning (*defence in depth*). For instance, a computer that is linked to the Internet should not solely rely on firewall [12, p. 9]. It is even more crucial when the computer runs services.

Code fragment 3.1 leaks several important information to an attacker. By knowing the database and table names it is possible to try database related attacks like SQL-injection to gain higher level access into the system. The path name reveals used operating system which is in this case Windows. By examining the stack trace (see Appendix) it reveals used programs and platforms (MySQL and ColdFusion). The previous information might be enough for the malicious attacker to penetrate into the system and/or crash the site.

An appropriate way to handle debug and system messages is to write errors to a log file instead showing them to users [12, pp. 35-41].

[2] *Root* user has the highest privilege. Also, ID number 0 refers to root.

```
You have an error in your SQL syntax; check the manual that corresponds to
your MySQL server version for the right syntax to use near '

and afdeling.provincie in (0,3) order by afdeling.sort' at line 3

The error occurred in D:\websites\kicker\content\Uitslagen\index.cfm: line 32

30 : select distinct(afdeling.afdelingid) as afdelingid, afdeling.afdeling
      as afdeling
31 : from reeks left join afdeling on (reeks.afdeling =
      afdeling.afdelingid)
32 : where seizoen = #qGetseizoen.maxseizoen# and afdeling.provincie in
      (0,#regio#) order by afdeling.sort;
33 : </cfquery>
34 : <cfoutput>

SQLSTATE 42000
SQL select distinct(afdeling.afdelingid) as afdelingid,
                afdeling.afdeling as afdeling from reeks left join afdeling on
                (reeks.afdeling = afdeling.afdelingid) where seizoen = and
                afdeling.provincie in (0,3) order by afdeling.sort;

VENDORERRORCODE 1064
DATASOURCE kickersql
```

Code fragment 3.1: An example presents an inappropriate way to handle errors.

The previous case happened to the author in the particular site. After reporting misconfiguration to the site administrator it had no effect nor the administrator replied to an email. A good example how security is seen as a low priority task for the administrator.

3.2 Compiler and language based safety methods

Machine language is the lowest language that computer "understands". It is hard to read and understand for human since it does not obey structure of a human language nor have similar lexical or syntactical expressions. *A symbolic machine language* remedies this shortage. By using the symbolic machine language multiplication operating is expressed as MUL [20, pp. 1-2]. Syntactically it is closer to *natural language*, though it the differs lexically a

lot.

Computer languages can be split into two different categories, *low level languages* and *high level languages*. The previous paragraph discussed about the machine language and the symbolic machine language. Both are part of the low level language category. These languages have a strict grammars. Vice versa, languages that human use for communications are more subtle, ambiguous, loosely defined and might vary syntactically a lot. As low level languages have a simple instruction set, it is hard to accomplish highly abstract tasks like implementing a B-Tree data structure. For this purpose there are higher level languages that are more subtle as compared on its predecessors, low level languages.

In a design process one should carefully choose the implementation language. The language that guarantees *type safeness* [13, p. 20], *integrated sandboxing* and is pointer free, should be always preferred. By using the language that implements type safeness it is impossible to mix strings and integers on the following way: $2 + "\#"$.

Misuse possibility exists if the language offers memory operations via pointers [13, p. 21]. Therefore pointer free (*memory safe*) languages should be preferred. Occasionally program has to be implemented in a language that does not fulfill these security needs. In these cases, functions that are boundary aware should be preferred and function return values should be verified in the case of errors. Also, function return values should always be checked against errors if the language does not implement exception handling itself.

In low level language an attacker can craft an exploit that is highly dependent on machine architecture instructions and bypass higher level language limits and safety methods. Albeit this is an interesting topic, the main focus of this study is in the higher level languages.

Majority of the open source operating systems are implemented in C language because it has support for a symbolic low level language. It is possible to write hardware drivers and make certain functions faster in lower level implementations. C language's main strength is in its speed. The language has many functions that gain speed by omitting boundary checks and by accessing memory directly via pointers. Although the speed assumption is true in general cases, Miller and de Raadt [10, p. 2] pointed out a safe function

implementation that does not have critical speed regression.

One of the functions that lacks off the boundary check is *strcat*. strcat is used for string concatenation. A manual page for *strncat* regarding the Open Group Base Specification [19] is Code fragment 3.2:

SYNOPSIS

```
#include <string.h>

char *strncat(char *restrict s1, const char *restrict s2, size_t n);
```

DESCRIPTION

The strncat() function shall append not more than n bytes (a null byte and bytes that follow it are not appended) from the array pointed to by s2 to the end of the string pointed to by s1. The initial byte of s2 overwrites the null byte at the end of s1. A terminating null byte is always appended to the result. If copying takes place between objects that overlap, the behavior is undefined.

Code fragment 3.2: Unix manual page of the strncat (only the relevant parts are included).

Prototype of the *strncat* takes exactly three arguments: $s1$ (destination), $s2$ (source) and n (size). If a source buffer size is greater than or equal to a destination buffer, NUL-terminating string is omitted. To use *strncat* safely, Miller and de Raadt [10, pp. 1-2] encourage to always copy size of $n - 1$ and NUL-terminate string by hand, although in some rare cases the previous operation is exaggerated.

Miller and de Raadt [10, p. 2] also disclose the misuse of the *strcnat*. Particularly the size parameter is often thought to be the size of the destination buffer. Miller and de Raadt use the following formalization to clear this misconception:

> *Most importantly, this is not the size of the destination string itself, rather it is the amount of space available.*

As their last concern, Miller and de Raadt [10, p. 3] pointed out that the performance regression made by the boundary check is diminutive. During the time the paper was released (1996), computers had significantly lower performance. Today when CPUs have more than doubled their computing performance and can do two operations in a single instruction, this claim is considered outdated.

3.2.1 Automatic memory management

There are two types of variables: local and global variables. The local variable has visibility in a function block. The global variable can be accessed anywhere from the same file. Many languages have adopted a syntax where *scope* is defined as an area between the curly brackets, { }. Code fragment 3.3 depicts the previous situation.

```
1  variable A // Global variable
2
3  function() {
4          variable B // Local variable
5  }
6
7  main() {
8          print(A) // Since A is global, this works
9          print(B) // Does not work, since B is local under function()
10 }
```

Code fragment 3.3: Local and global variables.

When a program returns from a function any of the locally reserved variables cannot be accessed anymore (excluding global variables). Age of the variable is consequently as long as the program execution stays in the scope.

Exceptions exist depending on the chosen programming language. C, for example, has an exception if a local variable is declared as a pointer and it is returned by the function. Apart from the previous exception, all dynamically allocated memory should be freed before leaving the function.

However, the previous approach has certain drawbacks. If one forgets to release the dynamically allocated memory in the end of the function, memory stays allocated and reference to it is lost. In the case the allocated information is confidential there is possibility to leak a confidential information. Next time when the memory allocation occurs and an attacker is determined, it is possible to allocate the same chunk of the memory. In certain languages that does not provide automatic memory handling, memory can be allocated without first emptying the memory area. One of these languages is C and to be specific, its function *malloc* can be used for the purpose. With certain knowledge the attacker can alter the previously used memory block and read the confidential information. One possible defend against this kind of attacks is to patch malloc to force erase allocated memory blocks. Such functional-

ity is provided at least in OpenBSD malloc [8] (J and Z switches). Other approach would be to switch to a memory safe language.

Garbage collection takes care that all dynamically loaded chunks are freed afterwards and the previously mentioned leak cannot happen. Without garbage collection every dynamically allocated memory block must be released by the user. Omitting the deallocation operation makes it possible to have a memory leak.

3.3 Protecting the memory

There are several ways to protect the memory from the leaks and the buffer overflows. The main focus is in the *ProPolice* in this topic as it prevents different kinds of attacks as proven by Wilander and Kamkar [21, 13-14].

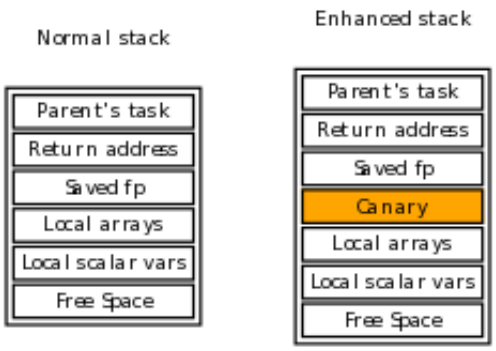


Figure 3.1: Structure of a computer memory stack and an enhanced memory stack.

Figure 3.1 shows an enhanced process memory layout that resides in a computer memory. Technique is known as ProPolice [7] and it protects against the *stack smashing attacks*. Protection works by inserting an extra information, canary (sometimes called as a cookie), into the stack. Canary is basically a random value. The canary is inserted into the each function call at the compile time [1, pp. 10-12]. If the canary changes during the execution time, program terminates.

Furthermore, ProPolice reorders the stack in a way that the flags and the pointers are placed lower in the stack. If overflow occurs, it likely first overwrites the canary. It makes harder to overwrite flags and pointers since

the canary changes are noticed. ProPolice has moderately low regression to performance since it is only about 1.3%.

Even the compiler would use canaries to protect against stack smashing problem, there are still possibility to pursue the attack. To make buffer overflow attacks even harder to gain any benefit, OpenBSD [11] operating system introduces the following methods [1, p. 16]: randomizing *ld.so* [3] memory load locations and orders, as well as randomizing *mmap* and malloc calls.

Other widely used approach against buffer overflow attacks is to use *NX-bit*. Abbreviation NX-bit stands for Non Execute Bit. It works by denying *write right* and *execute right* in the memory stack simultaneously. Its importance have been seen so crucial that modern processor manufacturers have implemented in a hardware level. Although it is widely used and works in general cases, it can be circumvented (Mastropaolo introduced a proof of concept attack [9]).

3.4 Access control methods

Many different access control methods exist and these can be tedious to find appropriate use for each application. It should be clear for what purposes access control are going to be used. Some environments work perfectly with DAC, whereas other environments demands more fine-grained approach and control for the information flow. On the latter cases role based access control (RBAC) methods like MAC can be considered.

Some of the methods are too complicated to be used consistently. ACL is a good example of ambiguous functionality where capability setting is not straightforward. Files that have many distinct rules come quickly unmaintainable.

An another example is from the Windows and the Macintosh OS X world where MAC restrictions (on by default) displease many regular users by not letting complete their tasks. After downloading a file from the Internet a dialog asks whether program should be executed. More similar pop ups will follow and screen is filled up with questions − just to complete a simple task. This leads to the situation where users turn off the MAC based access control and are again vulnerable to malicious applications.

[3] Run-time link-editor

3.4.1 Sandboxes and chroot

Sandboxing is an important factor when isolating running services or processes. Well designed sandbox implementation mitigates compromising the complete system in the case of malfunction of a program.

The chroot (*change root*) changes a given directory visibility. If a directory */home/user*1 is influenced by chroot, it points to the /, which equal for the root directory.

There are few drawbacks when chrooting a process. All the needed runtime dependencies must be copied inside the chrooted directory. Since */home/user*1 is now seen as /, file system hierarchy must be imitated inside the chrooted environment. The references to an object that resides outside the chroot does not work. For instance, soft links pointing out of the chroot are superfluous. If the process must run with root privileges, chroot renders useless [15], since root can always escape from the chroot environment.

Example 3.1. If the operating system updates *libc* and the chrooted program is written in C, all the dependent libraries must be updated manually to correspond non chroot system library versions. This step can be omitted if system libc update in chroot is neglected. Previous method is righteous if system is not security critical or the program does not have any vital operations. Program *ldd* (list dynamic object dependencies) is a helpful tool when replicating the program into the chroot environment.

3.4.2 Privilege separation

In a basic non-secure environment clients interacts directly with the privileged process. This poses system to a danger since malfunction can leak higher privileges or grant access to confidential information. Since it is not always possible to drop higher privileges, Niels et al. [14] implemented privilege separation. They propose isolation between the interacting clients and the server processes. By this way clients interact with non privileged process that have been forked from the privileged parent process. Privileged parent process can be needed for crucial tasks, like binding the listening address in low ports [4]. Splitting the processes makes program more resilient against attacks since malfunction only affects to the forked children. A mock-up in

[4] $Lowport < 1024$

Figure 3.2 shows how communication is made when a client from the Internet requests a HTTP-page.

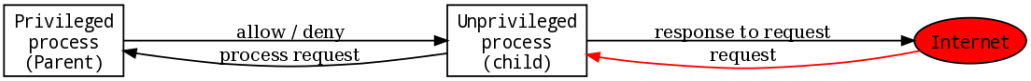


Figure 3.2: Communication negotiation between the Internet client and the service that have been privilege separated.

4 Conclusion

Completely secure systems do not exist. Users are and will always be the weakest link what comes to the computer security. Users write the software, develop the hardware and use the computers – appropriately or inappropriately. Well designed security environment does not guarantee protection against the all plausible flaws that exists or will exist but makes system penetration and exploitation harder. Even the safest system cannot be left without updates after the initial setup has been completed on a safe manner. Security is race against the time where the only hope is a incompetence of a malicious attacker. Many of the introduced methods offer safer way to execute third party programs, hide classified information, sustain integrity or make service fail on a safe manner.

As the conclusion to the language based security, several badly written application programming interfaces (API) exists and are used daily. Also, as proven on above, many of them are ambiguous and hard to use consistently. A leap in a security is to substitute all unsafe functions with the boundary aware functions and use APIs that have consistent interface.

If the security is the main concern, performance regression is inevitable, since values must be checked against the different safety limits. Only the magnitude of regression varies.

References

[1] Advances in OpenBSD. <`http://www.openbsd.org/papers/csw03/index.html`> (accessed: 19.12.2011).

[2] CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow') <http://cwe.mitre.org/top25/#CWE-120> (accessed 13.12.2011).

[3] Evans D. and Larochelle D., Improving Security Using Extensible Lightweight Static Analysis, *IEEE Software*, **19** (2002), 42-51.

[4] The Free Dictionary. <http://encyclopedia2.thefreedictionary.com/Keep+It+Simple%2c+Stupid>, (accessed 20.12.2011).

[5] Grünbacher A., POSIX Access Control Lists on Linux. <http://www.suse.de/~agruen/acl/linux-acls/online/> (accessed 19.11.2011).

[6] Harrison M. A., Ruzzo W. L., Ullman J. D., Protection in Operating Systems, *CACM* **19** (1976), 461-471.

[7] IBM Research. 2001. GCC extension for protecting applications from stack-smashing attacks. <http://www.trl.ibm.com/projects/security/ssp/> (accessed 10.11.2011).

[8] malloc, calloc, realloc, free, cfree - memory allocation and deallocation. <http://www.openbsd.org/cgi-bin/man.cgi?query=malloc&apropos=0&sektion=0&manpath=OpenBSD+Current&arch=i386&format=html> (accessed 15.12.2011).

[9] Mastropaolo M., Buffer overflow attacks bypassing DEP (NX/XD bits) - part 2 : Code injection. <http://www.mastropaolo.com/2005/06/05/buffer-overflow-attacks-bypassing-dep-nxxd-bits-part-2-code-injection/> (accessed 14.12.2011).

[10] Miller T. C. and de Raadt T. strlcpy and strlcat - Consistent, Safe,String Copy and Concatenation. In: *USENIX Conference -99*. Monterey, California. <http://openbsd.org/papers/strlcpy-paper.pdf> (accessed 02.10.2011).

[11] OpenBSD operating system. <http://www.openbsd.org> (accessed: 04.11.2011).

[12] Poll E. 2011a. Software security lecture notes. Lecture of design principles. The Kerckhoffs institute, Radboud. Nijmegen.

[13] Poll E. 2011b. Software security lecture notes. Lecture of language based security 1. The Kerckhoffs institute, Radboud. Nijmegen.

[14] Provos N., Fiedl M., Honeyman P. Preventing Privilege Escalation. In: *Proceedings of the 12th USENIX Security Symposium.* Washington, DC, 2003.

[15] Puffy at work - Code right and secure, The OpenBDS way. <`http://quigon.bsws.de/papers/2010/bsdcan/mgp00046.html`> (accessed 14.11.2011).

[16] de Raadt T., Discussion on buffer overflow prevention issues. <`http://openbsd.org/papers/csw03/mgp00015.html`> (accessed 12.12.2011).

[17] Splint - Annotation-Assisted Lightweight Static Checking. <`http://www.splint.org`> (accessed: 02.12.2011).

[18] TCP Wrappers <`http://www.softpanorama.org/Net/Network_security/TCP_wrappers/index.shtml`> (accessed 18.10.2011).

[19] The open group base specifications issue 6. <`http://pubs.opengroup.org/onlinepubs/009604599/functions/strncat.html`> (accessed: 11.12.2011).

[20] Tremblay J-P and Sorenson P. G., *Theory and Practice of Compiler Writing*, Global Media, Hyderabad, India, 2008.

[21] Wilander J. and Kamkar M., A Comparison of Publicly Available Tools for Dynamic Buffer Overflow Prevention, Linköpings universitet.

[22] WU-FTPD Development Group. <`http://wu-ftpd.therockgarden.ca`> (accessed: 02.12.2011).

[23] Zannone N. 2010-2011a. Distributed trust management lecture notes. Department of Mathematics and Computer Science. Eindhoven University of Technology. <`https://svn.win.tue.nl/viewvc/security_public/teaching/dtm/Slides/old%20slides/2010-2011/`> (accessed 18.12.2011).

[24] Zannone N. 2010-2011b. Distributed trust management lecture notes. Lecture of lattice based access models 1. Department of Mathematics and Computer Science. Eindhoven University of Technology. <https://svn.win.tue.nl/viewvc/security_public/teaching/dtm/Slides/old%20slides/2010-2011/03-Lattice-based%20Access%20Control%20Models%201.pdf?view=log> (accessed: 18.12.2011).

APPENDIX I

Stack trace regarding the inappropriately configured system.

```
com.mysql.jdbc.exceptions.MySQLSyntaxErrorException: You have an error in your
SQL syntax; check the manual that corresponds to your MySQL server version for
the right syntax to use near 'and afdeling.provincie in (0,3) order by
afdeling.sort' at line 3
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:936)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:2941)
        at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:1623)
        at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:1715)
        at com.mysql.jdbc.Connection.execSQL(Connection.java:3243)
        at com.mysql.jdbc.Connection.execSQL(Connection.java:3172)
        at com.mysql.jdbc.Statement.execute(Statement.java:706)
        at com.mysql.jdbc.Statement.execute(Statement.java:783)
        at coldfusion.server.j2ee.sql.JRunStatement.execute(JRunStatement.java:348)
        at coldfusion.sql.Executive.executeQuery(Executive.java:1210)
        at coldfusion.sql.Executive.executeQuery(Executive.java:1008)
        at coldfusion.sql.Executive.executeQuery(Executive.java:939)
        at coldfusion.sql.SqlImpl.execute(SqlImpl.java:325)
        at coldfusion.tagext.sql.QueryTag.executeQuery(QueryTag.java:831)
        at coldfusion.tagext.sql.QueryTag.doEndTag(QueryTag.java:521)
        at cfindex2ecfm893406114.runPage(D:\websites\kicker\content\Uitslagen\index.cfm:32)
        at coldfusion.runtime.CfJspPage.invoke(CfJspPage.java:192)
        at coldfusion.tagext.lang.IncludeTag.doStartTag(IncludeTag.java:366)
        at coldfusion.filter.CfincludeFilter.invoke(CfincludeFilter.java:65)
        at coldfusion.filter.ApplicationFilter.invoke(ApplicationFilter.java:279)
        at coldfusion.filter.RequestMonitorFilter.invoke(RequestMonitorFilter.java:48)
        at coldfusion.filter.MonitoringFilter.invoke(MonitoringFilter.java:40)
        at coldfusion.filter.PathFilter.invoke(PathFilter.java:86)
        at coldfusion.filter.ExceptionFilter.invoke(ExceptionFilter.java:70)
        at coldfusion.filter.ClientScopePersistenceFilter.invoke(ClientScopePersistenceFilter.java:28)
        at coldfusion.filter.BrowserFilter.invoke(BrowserFilter.java:38)
        at coldfusion.filter.NoCacheFilter.invoke(NoCacheFilter.java:46)
        at coldfusion.filter.GlobalsFilter.invoke(GlobalsFilter.java:38)
        at coldfusion.filter.DatasourceFilter.invoke(DatasourceFilter.java:22)
        at coldfusion.CfmServlet.service(CfmServlet.java:175)
        at coldfusion.bootstrap.BootstrapServlet.service(BootstrapServlet.java:89)
        at jrun.servlet.FilterChain.doFilter(FilterChain.java:86)
        at coldfusion.monitor.event.MonitoringServletFilter.doFilter(MonitoringServletFilter.java:42)
```

```
at coldfusion.bootstrap.BootstrapFilter.doFilter(BootstrapFilter.java:46)
at jrun.servlet.FilterChain.doFilter(FilterChain.java:94)
at jrun.servlet.FilterChain.service(FilterChain.java:101)
at jrun.servlet.ServletInvoker.invoke(ServletInvoker.java:106)
at jrun.servlet.JRunInvokerChain.invokeNext(JRunInvokerChain.java:42)
at jrun.servlet.JRunRequestDispatcher.invoke(JRunRequestDispatcher.java:284)
at jrun.servlet.ServletEngineService.dispatch(ServletEngineService.java:543)
at jrun.servlet.jrpp.JRunProxyService.invokeRunnable(JRunProxyService.java:203)
at jrunx.scheduler.ThreadPool$DownstreamMetrics.invokeRunnable(ThreadPool.java:320)
at jrunx.scheduler.ThreadPool$ThreadThrottle.invokeRunnable(ThreadPool.java:428)
at jrunx.scheduler.ThreadPool$UpstreamMetrics.invokeRunnable(ThreadPool.java:266)
at jrunx.scheduler.WorkerThread.run(WorkerThread.java:66)
```

No-Limit Texas Hold'emia pelaavat pokeriagentit

Esko Vankka

Tiivistelmä.
Pokeri tarjoaa tekoälytutkimukselle monia haasteita, jotka ovat tärkeä osa myös todellisen elämän päätöksentekoa. Pokerin päätöksentekotilanteille on tyypillistä useat keskenään kilpailevat osapuolet, joilla kullakin on omat tavoitteensa, epätäydellinen ja epäluotettava informaatio tilanteesta, riskien hallinta sekä tarkoituksellinen harhaanjohtaminen. Kahden pelaajan limit-pokerissa on jo onnistuttu kehittämään tekoälyohjelma, joka pystyy voittamaan ammattilaispelaajankin. Tässä tutkielmassa tarkastelen tutkimuksen tämän hetkistä tilaa esittelemällä Tartanianin, joka on Carnegie Mellon yliopiston kehittämä No-Limit Texas Hold'em agentti. Lisäksi esittelen lyhyesti muutamia muita pokeriagentteja, joissa on käytetty erilaisia lähestymistapoja ongelman ratkaisemiseksi.

Avainsanat ja -sanonnat: tekoäly, pokeri
CR-luokat: I.2.1

1. Johdanto

Laajat epätäydellisen informaation pelit tarjoavat tekoälytutkimukselle tutkimusalueen, johon liittyvät haasteet ovat läsnä myös todellisen elämän päätöksentekotilanteille. Pokeria pelaava agentti joutuu tekemään nopeita päätöksiä tilanteissa, joissa kaikki faktat eivät ole tiedossa ja osa käytettävissä olevasta tiedosta on epäluotettavaa. Tilanteen epävarmuutta lisäävät keskenään kilpailevat osapuolet, joilla kullakin on omat tavoitteensa ja jotka nimenomaan pyrkivät johtamaan harhaan vastustajiaan ja käyttämään hyväksi heidän heikkouksiaan.

Pokeria pelaavien tekoälyohjelmien tutkimus on aiemmin keskittynyt lähinnä limit-pokeria pelaaviin agentteihin. Muutaman viime vuoden aikana on kuitenkin tutkimusten kohteeksi otettu myös no-limit -pokeri. Koska No-Limit Texas Hold'emissa lyöntien kokoa rajoittaa vain käytettävissä olevien pelimerkkien määrä, kasvaa pelipuun koko limit-pokeriin verrattuna moninkertaiseksi.

Esittelen aluksi No-Limit Texas Hold'emin säännöt seuraavassa luvussa. Luvussa kolme esittelen kahden hengen No-Limit Texas Hold'em -agentin nimeltä Tartanian. Tartanian on Carnegie Mellon yliopistossa kehitetty peliteoriapohjainen pokeriagentti, joka on usean vuoden ajan menestynyt hyvin kilpailuissa. Neljannessä luvussa esittelen lyhyesti pokeriagenttien kehittämisessä käytettyjä erilaisia lähestymistapoja.

2. Heads-Up No-Limit Texas Hold'em -pelin säännöt

Jokaisesta pokeripelistä on olemassa useita erilaisia variaatioita. Tarkastelemani agentit on suunniteltu pelaamaan kahden pelaajan Doyle's Game -muunnelmaa, joka eroaa tavallisesta Texas Hold'em -pelistä vain siten, että kummallakin pelaajalla on jokaisen jaon alussa 1000 pelimerkkiä. Koska pelaajilla olevien pelimerkkien määrä vaikuttaa hyvin paljon optimaaliseen strategiaan, tämä sääntö takaa sen, että pelaajat ovat jokaisen jaon alussa tasavertaisessa asemassa. [Gilpin et al., 2008] Samalla se helpottaa pokeriagentin suunnittelua, sillä näin on yksi muuttuja vähemmän huomioonotettavana.

2.1. Alkupanokset, panostusten koko ja pelivuorot

Ennen korttien jakoa molemmat pelaajat maksavat pakolliset alkupanokset, joita kutsutaan pieneksi ja isoksi blindiksi. Tarkastelemassani variantissa pieni blind on yhden pelimerkin ja iso blind kahden pelimerkin suuruinen. Pienen blindin maksaa pelaaja, joka on jakajan paikalla (eli 'napilla'). Hän toimii ensimmäisellä panostuskierroksella ensimmäisenä ja muilla panostuskierroksilla viimeisenä. Toinen pelaaja maksaa ison blindin, toimii ensimmäisellä panostuskierroksella viimeisenä ja muilla panostuskierroksilla ensimmäisenä.

Kullakin panostuskierroksella ensimmäisen panostuksen täytyy olla vähintään ison blindin, eli tässä variantissa kahden pelimerkin, kokoinen. Korotuksen tulee olla aina vähintään viimeisimmän samalla kierroksella tehdyn panostuksen tai korotuksen kokoinen. Pelaajat voivat koska tahansa mennä all-in, eli panostaa kaikki loput hallussaan olevat merkit. [Miller, 2005]

2.2. Pre-Flop

Alkupanosten maksamisen jälkeen kummallekin pelaajalle jaetaan kaksi korttia 52 kortin pakasta. Näitä sanotaan taskukorteiksi. Ne jaetaan kuvapuoli alaspäin ja ovat vain pelaajan itsensä tiedossa.

Pienellä blindillä oleva pelaaja aloittaa. Hän voi luovuttaa, maksaa yhden merkin lisää (täydentää blindin) tai korottaa. Jos hän luovuttaa, niin isolla blindillä oleva pelaaja voittaa alkupanokset. Jos hän maksaa tai korottaa, siirtyy päätösvuoro isolle blindille.

Jos vastustaja on maksanut, voi isolla blindillä oleva pelaaja checkata, jolloin siirrytään seuraavalle kierrokselle, tai hän voi korottaa, jolloin päätösvuoro siirtyy takaisin vastustajalle (joka taas vuorollaan voi luovuttaa, maksaa tai korottaa). Jos pieni blind on korottanut, voi pelaaja isolta blindilta luovuttaa, maksaa tai korottaa lisää. Näin jatketaan, kunnes toinen pelaajista luovuttaa jaon (jolloin hänen vastustajansa voittaa potin ja siirrytään seuraavaan jakoon), tai maksaa viimeisimmän korotuksen (jolloin siirrytään seuraavalle kierrokselle, eli flopille). [Miller, 2005]

2.3. Flop, turn ja river

Pöytään jaetaan kolme yhteistä korttia kuvapuoli ylöspäin. Tämän jälkeen seuraa uusi panostuskierros. Isolla blindilla oleva pelaaja aloittaa ja panostus etenee samalla tavoin kuin edelliselläkin kierroksella. Ensimmäisen panostuksen täytyy olla vähintään ison blindin kokoinen, eli tässä variantissa kaksi pelimerkkiä. Jos pelaajat ovat all-in, ei panostuksia enää tehdä, vaan loput yhteiskortit jaetaan pöytään ja katsotaan, kummalla on parempi käsi.

Turnilla jaetaan pöytään yksi yhteinen kortti lisää ja panostuskierros pelataan samalla tavoin kuin flop.

Riverillä jaetaan pöytään viimeinen yhteinen kortti ja panostuskierros pelataan jälleen kuten kahdella edellisellä kierroksella. [Miller, 2005]

2.4. Showdown

Jos kumpikaan pelaaja ei ole luovuttanut riverin jälkeen, katsotaan kumpi voittaa. Kumpikin pelaaja muodostaa parhaan mahdollisen pokerikäden käyttäen yhteensä viittä korttia kahdesta taskukortistaan ja pöydässä olevista viidestä yhteisestä kortista. Paremman pokerikäden omaava pelaaja voittaa potin. Jos kädet ovat samanarvoiset, jaetaan potti puoliksi. Tämän jälkeen siirrytään seuraavaan jakoon. [Miller, 2005]

3. Tartanian

Tartanian on Carnegie Mellon yliopistossa kehitetty kahden hengen no-limit Texas Hold'em -pokeria pelaava agentti. Tartanianin eri versiot ovat menestyneet hyvin jokavuotisessa Computer Poker Competition -kilpailussa saavuttaen toisen sijan vuonna 2007 ja voittaen yhden No-limit -sarjoista vuonna 2010 [ACPC, 2011]. Tartanian käyttää strategioidensa laatimiseen peliteoriaa yrittäen löytää strategian, joka olisi mahdollisimman lähellä Nashin tasapainoa (Nash equilibrium). Tasapainon etsimiseen Tartanian käyttää uutta tekniikkaa, joka automaattisesti generoi tasapainonetsimiseen käytettävän algoritmin lähdekoodin. Pelipuun valtavan koon hallitsemiseksi Tartanian käyttää lyöntimallien diskretisoimista ja potentiaalin huomioon ottavia, automaattisia abstraktioalgoritmeja. Seuraavissa kohdissa esittelen tarkemmin näitä menetelmiä. [Gilpin et al., 2008]

3.1. Diskretisoitu lyöntimalli

Limit Hold'emissa pelaajalla on aina vuorollaan korkeintaan kolme vaihtoehtoa. Pelaaja voi luovuttaa, checkata/maksaa tai lyödä/korottaa. No-limit Hold'emissa pelaaja voi lisäksi valita lyömänsä tai korottamansa summan vapaasti aina käytössään olevien pelimerkkien kokonaismäärään asti. Tämä vai-

kuttaa merkittävästi optimaaliseen strategiaan ja lisää pelissä esiintyvien tilojen määrää moninkertaisesti. Tilojen määrä olisi rajaton, jos sallittaisiin pelimerkin osien käyttö lyöntisummaa määrättäessä. Yleensä merkin osien käyttö on kuitenkin säännöissä kielletty ja joka tapauksessa pelimerkin osien merkitys strategian kannalta olisi niin pieni, että ne voitaisiin mielestäni huoletta pyöristää lähimpään kokonaislukuun. Täydellinen pelipuu on no-limitissä silti huomattavasti suurempi kuin limitissä. Jos pienessä blindissa olevalla pelaajalla on käytössään 1000 pelimerkkiä jaon alussa, hän voi luovuttaa, maksaa tai korottaa minkä tahansa summan väliltä 4-1000 pelimerkkiä. Hän voi siis ensimmäisellä vuorollaan valita toimintonsa 999 eri vaihtoehdosta. Vastaavankokoisia päätöksiä esiintyy myös muualla pelipuussa. Tämän seurauksena kahden hengen no-limit Texas Hold'emin pelipuu kasvaa noin 10^{71} solmuun, kun limit -version pelipuu sisältää 10^{18} solmua. [Gilpin et al., 2008]

Pelipuun valtavan koon hallitsemiseksi Tartanianissa käytetään diskretisoitua lyöntimallia. Lyöntimallin diskretisointi koostuu kahdesta osasta: mallissa sallittujen lyöntien kokojen valinnasta ja todellisen pelin tapahtumien kuvaamisesta takaisin abstraktin pelin tapahtumiksi.

3.1.1. Lyöntimalli

Ihmispelaajat käyttävät useimmiten tietynkokoisia lyöntejä suhteessa sen hetkiseen pottiin. Yleisimmät lyönnit ovat 50 %, 75 % ja 100 % potin koosta sekä all-in.

Potin puolikkaat ovat hyviä arvolyönteinä, sillä saadessaan 3:1 pottikertoimet on maksaminen houkutteleva vaihtoehto vastustajalle. Ne ovat hyviä myös bluffeina, sillä niiden täytyy toimia vain kerran kolmesta ollakseen tuottoisia.

Potin kokoiset lyönnit ovat hyviä silloin, kun pelaaja uskoo olevansa johdossa sillä hetkellä, eikä halua antaa vastustajalle mahdollisuutta yrittää parantaa kättään halvalla myöhemmillä kierroksilla. Pottia suuremmat lyönnit ovat yleensä turhia.

All-in -lyönti on yleisesti ottaen kannattava vain, jos potin koko on suuri verrattuna pelaajien jäljellä olevien pelimerkkien määrään. Jos vastustaja maksaa all-in -lyönnin potin ollessa pieni, on hänellä hyvin suurella todennäköisyydellä parempi käsi. Jos hänellä on huonompi käsi, hän ei yleensä maksa, eikä lyönnistä siis ole mitään varsinaista hyötyä. Joissakin tilanteissa, potin ollessa suuri, all-in -lyönti saattaa kuitenkin olla kannattava.

Lyönnit, jotka ovat hyvin pieniä suhteessa potin kokoon, antavat vastustajalle niin hyvät kertoimet, että hänen kannattaa maksaa lähes millä tahansa kädellä. Siten hänen maksunsa ei anna minkäänlaista lisäinformaatiota hänen kädestään. Liian pieni lyönti ei myöskään tuo paljoa lisäarvoa pelaajalle, jolla on parempi käsi.

Kun pelaajan jäljellä olevien merkkien määrä on pieni suhteessa pottiin, hän on sitoutunut pottiin. Tämä tarkoittaa sitä, että pelaaja saa niin hyvät kertoimet maksaessaan lopuilla merkeillään, että hänen ei kannata olla maksamatta millään korteilla. Toisin sanoen järkevä pelaaja, joka on sitoutunut pottiin, on käytännössä samassa tilanteessa kuin jos hän olisi saman tien all-in. Siten sitoutumiseen johtavat lyönnit ovat itse asiassa lähes tarpeettomia. Tämän vuoksi Tartanianin käyttämässä lyöntimallissa ei käytetä lyöntejä, jotka johtaisivat pottiin sitoutumiseen. [Gilpin et al., 2008]

Teoriassa pelaajat voisivat lyödä ja korottaa vuorotellen useita kertoja kierroksen aikana. Useimmiten lyöntisarjat kuitenkin koostuvat vain yhdestä tai kahdesta lyönnistä. Tämän vuoksi Tartanianin lyöntimallissa rajoitettiin lyöntien määrä korkeintaan kolmeen kierrosta kohti. Myöhemmin kehittäjät kuitenkin huomasivat, että rajattoman lyöntimäärän mahdollistaminen yhdessä puolen potin kokoisen minimilyönnin kanssa lisäisi mallin kokoa vain 15%, joten myöhemmissä versioissa he aikovat poistaa tämän rajoituksen. [Gilpin et al., 2008]

Yllä olevien seikkojen pohjalta Tartanianiin suunniteltiin lyöntimalli, joka sallii seuraavat toiminnot:

1. Pelaajilla on aina mahdollisuus mennä all-in.

2. Kun yhtään lyöntiä ei vielä ole tehty kyseisellä lyöntikierroksella, on pelaajalla mahdollisuus checkata, lyödä 50% potista, lyödä 100% potista tai mennä all-in.

3. Vastustajan lyötyä on pelaajalla mahdollisuus luovuttaa, maksaa, lyödä potin verran tai mennä all-in.

4. Jos missään vaiheessa tietty lyöntikoko on yli puolet pelaajan merkeistä, poistetaan kyseinen lyöntikoko lyöntimallista.

5. Kullakin lyöntikierroksella pelaajan on sallittua tehdä korkeintaan kolme lyöntiä. [Gilpin et al., 2008]

Erilaisten lyöntikokojen rajoittaminen näin rajusti on selkeä strateginen heikkous, mutta pelipuun pienenemisestä on vastaavasti suurta hyötyä.

3.1.2. Takaisinkuvaus

Todellisessa pelissä vastustaja ei luultavastikaan tyydy käyttämään vain tiettyyn lyöntimalliin kuuluvia lyöntejä. Tarvitaan malli, jonka mukaisesti vastustajan lyönnit kuvataan abstraktiin malliin sopiviksi. Mallin sisältäessä vain puolen potin ja koko potin kokoisia lyöntejä, täytyy esimerkiksi vastustajan tekemä ¾ potin kokoinen lyönti tulkita vastaamaan jotakin malliin kuuluvaa lyöntikokoa.

Yksi mahdollisuus on kuvata tapahtumat lähimpään mallista löytyvään tapahtumaan. Esimerkiksi jos vastustaja lyö 4/5 potista, voidaan sitä käsitellä sa-

moin kuin jos hän olisi lyönyt täyden potillisen. Tämä tapa on kuitenkin haavoittuva. Esimerkiksi, jos molemmilla pelaajilla on 1000 pelimerkkiä jäljellä ja potti on 5 merkin kokoinen, niin 500 merkin lyönti tulkittaisiin potin kokoiseksi lyönniksi. Tällöin mallimme suosisi maksamista aivan liian huonoilla korteilla, sillä 500 merkin lyönti suhteessa 2 merkin pottiin on niin suuri, että sen todellinen merkitys on sama kuin all-in.

Toinen mahdollisuus olisi arpoa kahdesta mallin antamasta vaihtoehdosta toinen siten, että valituksi tuleminen on sitä todennäköisempää, mitä lähempänä mallin rajapiste on. Tälläkin menetelmällä päädyttäisiin kuitenkin yllämainitussa tapauksessa edelleen väärään ratkaisuun noin puolet kerroista.

Tartanian käyttää absoluuttisen etäisyyden sijaan suhteellista etäisyyttä valitessaan lähimmän tapahtuman. Esimerkiksi jos vastustaja tekee lyönnin, jonka koko on c merkkiä ja lähimmät vastineet mallissa ovat d1 ja d2 (d1 < c < d2), niin verrataan suhteita c/d1 ja d2/c ja valitaan niistä pienempi. Yllä olevassa esimerkissä saadaan nyt c/d1 = 500/5 = 100 ja d2/c = 1000/500 = 2. Tällä menetelmällä siis tulkitaan tämä lyönti aivan oikein all-in -lyönnin vahvuiseksi. [Gilpin et al., 2008]

3.2. Automaattinen korttien abstraktio

Panostustapahtumien abstrahoimisen lisäksi täytyy abstrahoida myös sattumatapahtumat eli korttien jakaminen. Tämä osa-alue on pitkälti sama limit ja no-limit -versioissa ja sitä on tutkittu aiemminkin [Gilpin et al., 2007c]. Tartanian käyttää hyväkseen näitä aiemmin kehitettyjä tekniikoita.

GameShrink-algoritmi kehitettiin tätä tarkoitusta varten. Algoritmi perustuu siihen, että tietyt pelipuun solmut ovat strategisesti täysin yhtäläisiä. Esimerkiksi Texas Hold'emissa pelaajalle voidaan jakaa ässäpari kuudella eri tavalla, mutta ne kaikki ovat strategisesti identtisiä tilanteita. Samoin jos flopilla pelaajalla on setti (kolme samaa korttia, joista kaksi on kädessä ja yksi pöydässä), ei korttien maalla juurikaan ole strategista merkitystä, elleivät pöydässä olevat kortit ole kaikki keskenään samaa maata. GameShrink-algoritmi etsii tällaiset strategiset symmetriatilanteet ja yhdistää ne keskenään. Näin saadaan pelipuuta kutistettua huomattavasti ja tasapainoanalyysi voidaan suorittaa saadulle abstraktille pelille. Tuloksena saatu tasapaino on identtinen alkuperäisen pelin tasapainon kanssa, joten abstraktio on häviötön. Texas Hold'em on kuitenkin liian laaja peli ratkaistavaksi tällä häviöttömällä algoritmilla, ja niinpä Albertan yliopiston Limit Texas Hold'em -tekoälyohjelma, GS1, käyttää hieman muunnettua versiota, joka ei ole häviötön. [Gilpin et al., 2008]

GS1:ssä käytetyssä algoritmissa havaittujen puutteiden vuoksi kehitettiin GS2:ta varten uusi automaattinen abstraktioalgoritmi, joka perustuu k:n keskiarvon klusterointiin ja kokonaislukuohjelmointiin [Gilpin et al., 2007b]. Algo-

ritmi laskee eri käsien voittomahdollisuudet ja jakaa kädet niiden perusteella ryhmiin (buckets). Algoritmin tarkkuutta voidaan lisätä ryhmien lukumäärä lisäämällä. Voittomahdollisuudet lasketaan käden absoluuttisen vahvuuden perusteella, eikä algoritmi ota huomioon sitä, että käden vahvuus muuttuu lyöntikierrokselta toiselle siirryttäessä. Algoritmi jättää huomiotta myös sen, että eri käsien vahvuus vaihtelee erilaisilla hakupoluilla. [Gilpin et al., 2008]

GS3:a varten algoritmista kehitettiin edelleen potentiaalitietoinen automaattinen abstraktioalgoritmi, jota Tartanian käyttää No-limit Texas Hold'emissa. Potentiaalitietoinen algoritmi ottaa käden absoluuttisen vahvuuden lisäksi huomioon myös käden (positiiviset tai negatiiviset) kehittymismahdollisuudet. Tartanianissa kädet jaetaan ensimmäisellä kierroksella 10 ryhmään, toisella kierroksella 150 ryhmään, kolmannella kierroksella 750 ryhmään ja neljännellä kierroksella 3750 ryhmään. Ryhmien määrä perustuu arvioihin siitä, kuinka laajan ongelman agentissa käytettävä tasapainonetsintäalgoritmi pystyy ratkaisemaan tarkasti ja hyväksyttävässä ajassa. [Gilpin et al., 2008]

3.3. Peliteoreettisen tasapainon laskeminen

Täydellinen Nash-tasapaino on tila, jossa yksikään pelaaja ei voi parantaa odotusarvoaan muuttamalla strategiaansa. Suhteellisen pienille peleille voidaan löytää Nash-tasapaino melko helposti lineaarisilla algoritmeilla. Pienten kahden hengen nollasummapelien mallintamiseen ja ratkaisemiseen käytetyt lineaariset algoritmit eivät kuitenkaan sovellu Texas Hold'emin kaltaisten laajojen pelien ratkaisemiseen. Tämän vuoksi Tartanian käyttää Carnegie Mellon yliopistossa kehitettyjä gradientteihin perustuvia algoritmeja, jotka skaalautuvat monta kertaluokkaa suurempiin peleihin kuin aiemmin oli mahdollista. [Gilpin et al., 2008]

Tartanianin käyttämät algoritmit yrittävät täydellisen Nash-tasapainon sijaan löytää e-tasapainon. Etsittäessä e-tasapainoa pyritään pääsemään niin lähelle täydellistä tasapainoa kuin mahdollista. Tässä e tarkoittaa suurinta mahdollista hyötyä, jonka pelaaja voi saavuttaa muuttamalla strategiaansa. Tavoitteena on siis kehittää strategioita, joissa e on mahdollisimman pieni. [Gilpin et al., 2008]

Tartaniania suuniteltaessa otettiin lähtökohdaksi Nesterovin [2005] EGT-algoritmi, joka on kehitetty e-tasapainon etsimiseen kahden hengen vuoropohjaisissa peleissä. Algoritmia parannettiin kehittämällä heuristiikka, joka nopeutti algoritmia huomattavasti säilyttäen kuitenkin algoritmin teoreettisen konvergenssin. Lisäksi mukaan liitettiin hyvin skaalautuva ja samanaikaistuva (parallelizeable) toteutus matriisi-vektori -tulon laskentaan. [Gilpin et al., 2007a]

Matriisi-vektori -tulon laskenta on EGT-algoritmissa eniten aikaa vievä osa. Pienille peleille, joiden strategia-avaruuden rakenne on suhteellisen yksinker-

tainen, kyseisen tulon laskenta-algoritmin koodi voidaan helposti kirjoittaa käsin. Suurille ja monimutkaisille peleille vaadittavat algoritmit sen sijaan ovat huomattavasti monimutkaisempia. Näiden koodien generoiminen käsin olisi vaikeata ja hidasta. Lisäksi työ pitäisi tehdä uudestaan jokaiselle pelille ja jokaiselle lyöntimallin diskretisoinnille. Tämän vuoksi Gilpin ja muut [2008] kehittivät tavan generoida kyseisen koodin automaattisesti XML-pohjaisen lyöntimallin kuvauksen perusteella.

4. Muita lähestymistapoja

Tässä luvussa esittelen vielä lyhyesti joitakin erilaisia lähestymistapoja no-limit -pokeriagenttien ohjelmoimiseen.

4.1. Tietopohjaiset järjestelmät

Tietopohjaisia järjestelmiä kehitettäessä sovellusalan asiantuntija avustaa järjestelmän suunnittelussa. Asiantuntijan tietämys aiheesta pyritään tallentamaan järjestelmään jossakin sellaisessa muodossa, että tekoälyohjelma pystyy käyttämään sitä. Tietopohjaiset lähestymistavat Texas Hold'em -agentteihin voidaan jakaa kahteen luokkaan: sääntöpohjaisiin ja kaavapohjaisiin järjestelmiin. [Rubin and Watson, 2011]

4.1.1. Sääntöpohjaiset järjestelmät

Sääntöpohjainen järjestelmä voi yksinkertaisimmillaan olla joukko if-lauseita, yksi if-lause kutakin mahdollisesti eteen tulevaa tilannetta kohti. If-lauseilla ei kuitenkaan yleensä suoraan päätetä esimerkiksi luovuttaa tai korottaa, vaan lauseen ehdon toteutuessa luodaan tilanteeseen sopiva todennäköisyys kolmikko (probability triple). Esimerkiksi jos pelaaja saa jaon alussa kaksi ässää, voisi luotu kolmikko olla Triple(0.0, 0.05, 0.95). Kolmikon luvut tarkoittavat sitä, kuinka usein pelaajan tulisi luovuttaa, maksaa ja korottaa. Tässä tapauksessa pelaajan ei tulisi koskaan luovuttaa, hänen tulisi maksaa 5% kerroista ja korottaa 95% kerroista. Lopullinen päätös tehdään sitten satunnaislukugeneraattorilla, joka on painotettu kolmikon luvuilla. [Rubin and Watson, 2011]

4.1.2. Kaavapohjaiset strategiat

Kaavapohjaiset järjestelmät ovat yleiskäyttöisempiä kuin sääntöpohjaiset järjestelmät. Kaavapohjaiselle järjestelmälle annetaan joukko tietoja pelin tilanteesta. Näistä tiedoista lasketaan sitten jollakin kaavalla todennäköisyyskolmikko, jonka pohjalta lopullinen lyöntipäätös satunnaisesti valitaan. Kaavalle annetut tiedot voivat olla esimerkiksi käden vahvuus ja pottikerroin. [Rubin and Watson, 2011]

Pottikerroin lasketaan kaavalla P = c/p+c, kun c on summa, joka pelaajan täytyy maksaa pysyäkseen pelissä ja p on potin koko kyseisellä hetkellä. Esimerkiksi jos potissa on 80 pelimerkkiä ja pelaajan pitää maksaa 20 merkkiä pysyäkseen mukana, on pottikerroin 0,20. Pottikerrointa käytetään apuna päätettäessä, kannattaako maksaa. Tässä tapauksessa pelaajalla täytyy olla yli 20% mahdollisuus voittaa käsi, jotta hänen kannattaa jatkaa. [Rubin and Watson, 2011]

Käden vahvuuden laskemiseen on kehitetty useita eri menetelmiä [Billings et al., 2002]. Yksi tapa on IHR (immediate hand rank), joka kuvaa käden välitöntä vahvuutta. IHR lasketaan käymällä läpi kaikki kädet, jotka vastustajalla voi kyseisessä tilanteessa olla ja katsomalla, miten pelaajan oma käsi pärjää kutakin mahdollista vastustajan kättä vastaan. Saatujen tulosten perusteella IHR lasketaan kaavalla IHR = (voitot + (tasapelit/2)) / (voitot + tasapelit + häviöt). IHR:n laskentaa voidaan parantaa painottamalla vastustajan mahdollisia kortteja sen mukaan, millä todennäköisyydellä vastustaja pelaa kyseiset kortit. [Rubin and Watson, 2011]

EHS (effective hand strength) ottaa käden vahvuuden laskemisessa huomioon tulevien korttien vaikutuksen käden vahvuuteen. EHS lasketaan kaavalla EHS = IHS + (1 - IHS) x Potentiaali, kun IHS on käden välitön vahvuus ja Potentiaali on todennäköisyys, jolla tällä hetkellä jäljessä oleva käsi kehittyy parhaaksi kädeksi jaon edetessä. [Rubin and Watson, 2011]

4.1.3. Tietopohjaisten järjestelmien ongelmia

Tietopohjaiset järjestelmät tarjoavat yksinkertaisen lähestymistavan pokeriagenttien kehittämiseen, mutta niissä on monia ongelmia. Tietomäärän lisääntyessä järjestelmästä tulee monimutkainen ja vaikeasti ylläpidettävä, minkä seurauksena järjestelmään saattaa eksyä ristiriitaista tai virheellistä tietoa. Tällainen järjestelmä on myös liian jäykkä, eikä pysty sopeutumaan vastustajan mahdollisesti muuttaessa strategiaansa. [Rubin and Watson, 2011]

McCurley [2009] on vertaillut tietopohjaista lähestymistapaa epätäydellisen informaation pelipuun käyttöön No-limit Hold'em -agentin toteuttamisessa. Hän testasi agentteja ihmispelaajia vastaan ja huomasi, että tietopohjainen lähestymistapa oli tappiollinen, kun taas epätäydellisen informaation pelipuuta käyttävä agentti oli voitollinen. [Rubin and Watson, 2011]

4.2. Peliteoreettiset tasapainomenetelmät

Etsittäessä parasta mahdollista pokeristrategiaa voidaan apuna käyttää peliteorian Nash-tasapainon käsitettä. Ajatellaan, että joukko pelaajia pelaa toisiaan vastaan kukin omalla strategiallaan. Nash-tasapaino on silloin sellainen joukko strategioita, että jokainen pelaaja käyttää optimaalista strategiaa kaikkien tois-

ten pelaajien käyttämiä strategioita vastaan. Toisin sanoen yksikään pelaaja ei voi parantaa voitto-osuuttaan strategiaansa muuttamalla. [Chen and Ankenman, 2006]

Nash -tasapainoa etsivät menetelmät tuottavat staattisia strategioita, joiden tarkoitus on minimoida vastustajan mahdollisuudet hyväksikäyttää strategian heikkouksia. Tällaiset strategiat minimoivat omat tappionsa, mutta eivät pysty havaitsemaan ja hyväksikäyttämään vastustajan heikkouksia. Peliteoreettisten menetelmien käyttö on eniten tutkittu lähestymistapa pokeriagentteja kehitettäessä. Tasapainomenetelmiä käyttävät agentit ovat myös viime vuosina pärjänneet parhaiten kilpailuissa. Teppo Salosen BluffBot [Salonen, 2011], Albertan yliopiston Hyperborean [Schnizlein, 2009] ja Carnegie Mellon yliopiston Tartanian ovat vuodesta toiseen olleet kolmen parhaan joukossa jokavuotisessa Computer Poker Competition -kilpailussa [ACPC, 2011].

Olen aiemmin tässä tutkielmassa esitellyt Tartanian-agenttia tarkemmin. BluffBot-agentti on mielenkiintoinen, sillä se on ilmeisesti yksityisen henkilön kehittämä ja pärjännyt hyvin kilpailuissa. Valitettavasti siitä ei ole saatavilla tarkkoja tietoja.

Albertan yliopiston Hyperboreanin eräs versio [Schnizlein, 2009] käyttää e-Nash -tasapainon laskemiseen CFR-algoritmia (Counterfactual regret minimisation). CFR on laajojen pelien tasapainon etsimiseen kehitetty iteratiivinen algoritmi, joka käsittelee yksittäisten pelitilanteiden sijasta tietojoukkoja, joihin pelitilanteet on jaettu. [Schnizlein, 2009]

4.3. Vastustajan heikkouksia hyväksikäyttävät vastastrategiat

Toisin kuin peliteoreettiseen tasapainoon perustuvat agentit, vastustajan heikkouksia hyväksikäyttävät agentit ovat valmiita hyväksymään joitakin heikkouksia omassa strategiassaan pystyäkseen ottamaan kaiken irti vastustajan heikkouksista. Tällaiset agentit pyrkivät siis korvaamaan parempia vastustajia vastaan kärsimänsä tappiot keräämällä suuremmat voitot heikommilta vastustajilta.

Vastustajan heikkouksia hyväksikäyttävät agentit käyttävät vastustajien mallinnusta (opponent modeling) kartoittaakseen vastustajan pelistrategiaa ja etsiäkseen siitä heikkouksia. Menetelmät voivat olla mukautuvia tai staattisia. Mukautuvat menetelmät käyttävät epätäydellisen informaation pelipuun hakualgoritmeja. Staattiset hyväksikäyttömenetelmät taas pohjautuvat peliteoreettisiin tasapainomenetelmiin, mutta ovat valmiita hyväksymään joitakin haavoittuvuuksia voidakseen hyväksikäyttää vastustajiaan paremmin. [Rubin and Watson, 2011]

McCurley [2009] käytti no-limit -agentissaan epätäydellisen informaation hakuja ja on-line-pokerisivustoilta kerätyillä käsihistorioilla koulutettua neuro-

verkkoa. Vuoden 2009 versio Albertan yliopiston Hyperborean-agentista taas käytti tekniikkaa, jossa Nash-tasapaino laskettiin pelissä, jossa vastustajan strategian kussakin pelipuun solmussa oletettiin olevan jokin tietty strategia tietyllä todennäköisyydellä [Johanson and Bowling, 2009].

4.4. Tapauspohjainen päättely

Tapauspohjainen päättely on oppimisalgoritmi, jossa tallennetaan ja ylläpidetään tapauksia ja niiden ratkaisuja. Etsittäessä ratkaisua käsillä olevaan tilanteeseen verrataan sitä aiemmin kohdattuihin tapauksiin ja valitaan niistä lähimmin käsillä olevaa tilannetta vastaavan tapauksen ratkaisu.

Aucklandin yliopistossa kehitetty SartreNL-agentti [Rubin and Watson, 2009] käyttää tapauspohjaista päättelyä menestyksekkäästi. Sartre koulutettiin käyttämällä parhaiten menestyneiden e-Nash -tasapainoon perustuvien agenttien käsihistorioita aiempien vuosien kilpailuista. Sartre saavutti toisen sijan vuoden 2011 Computer Poker Competition -kilpailun kahdessa eri no-limit-sarjassa [ACPC, 2011].

4.5. Evolutionääriset algoritmit ja neuroverkot

Evolutionäärisillä algoritmeilla pyritään kehittämään vahvoja agentteja automaattisesti. Menetelmässä agentit pelaavat toisiaan vastaan ja parhaiten pärjäävät otetaan mukaan seuraavaan sukupolveen, jossa niistä yhdistämällä tuotetaan jälkeläisiä. Joissakin menetelmissä eri populaation voivat kilpailla myös keskenään, mutta evoluutio on rajoitettu populaation sisälle. Nicolai ja Hilderman [2009] ovat tutkineet evolutionääristä menetelmää No-Limit Texas Hold'em agenttien kehittämisessä.

5. Yhteenveto

Pokeri on toiminut hyvin tekoälytutkimuksen kenttänä jo usean vuoden ajan. Pokeria pelaavan agentin tulee pystyä toimimaan tehokkasti vihamielisessä ympäristössä epätäydellisen informaation varassa. Tutkimus on aiemmin keskittynyt lähinnä limit-pokeriin, mutta viimeaikoina on ryhdytty tutkimaan myös no-limit -pokeria. No-limit Texas Hold'em tarjoaa uusia haasteita tekoälytutkimukselle, sillä sen pelipuu on huomattavasti suurempi kuin limit-variaatiossa.

Kilpailukykyisten pokeriagenttien kehittämiseksi on tutkittu useita erilaisia lähestymistapoja. Eniten on tutkittu peliteoreettiseen tasapainoon perustuvia agentteja ja ne ovatkin menestyneet kilpailuissa tasaisesti vuodesta toiseen. Vaihtoehtoisista lähestymistavoista ovat vastustajien heikkouksia hyväksikäyttävät strategiat sekä tapauspohjaiseen päättelyyn perustuvat agentit nousseet viime vuosina haastamaan puhtaasti peliteoreettisia menetelmiä.

Tähän mennessä tutkimus on keskittynyt lähinnä kahden hengen peleihin, joihin tässäkin tutkielmassa on keskitytty. Usean pelaajan pokeria on tutkittu verrattain vähän johtuen valtavan laajan pelipuun käsittelemisen vaatimasta laskentatehosta. Vaikka kehitystä on tapahtunut paljon, tarjoaa pokeri edelleen suuria haasteita tekoälytutkimukselle.

Viiteluettelo

[ACPC, 2011] ACPC, The Annual Computer Poker competition, http://www.computerpokercompetition.org/, checked 26.11.2011.

[Billings et al., 2002] D. Billings, A. Davidson, J. Schaeffer and D. Szafron, The Challenge of Poker. *Artificial Intelligence* **134**, 1-2 (Jan 2002), 201-240.

[Chen and Ankenman, 2006] Bill Chen and Jerrod Ankenman, *The Mathematics of Poker.* ConJelCO, 2006.

[Gilpin et al., 2007a] A. Gilpin, S. Hoda, J. Peña, and T. Sandholm, Gradient-based algorithms for finding Nash equilibria in extensive form games. In: Proc. of *3rd International Workshop on Internet and Network Economics*, WINE'07, 57–69.

[Gilpin et al., 2007b] A.Gilpin and T. Sandholm, Better automated abstraction techniques for imperfect information games. In: *Proc. of 6th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2007. AAMAS '07, 1176–1183.

[Gilpin et al., 2007c] A. Gilpin, T. Sandholm, and T. B. Sørensen, Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of Texas Hold'em poker. In: *Proc. of the National Conference on Artificial Intelligence*, 2007. AAAI'07, **1**, 50–57.

[Gilpin et al., 2008] Andrew Gilpin, Tuomas Sandholm, and Troels Bjerre Sørensen, A heads-up no-limit Texas Hold'em poker player: Discretized betting models and automatically generated equilibrium-finding programs. In: *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2008. AAMAS 2008, 911–918.

[Johanson and Bowling, 2009] M. Johanson and M. Bowling, Data biased robust counter strategies, In: *Proc. of Twelfth International Conference on Artificial Intelligence and Statistics*, 2009, AISTATS 2009, 264–271.

[McCurley, 2009] P. McCurley, An artificial intelligence agent for Texas hold'em poker, Undergraduate dissertation, University of Newcastle Upon Tyne, 2009.

[Miller, 2005] Ed Miller, *Getting Started in Hold'em.* Two Plus Two Publishing, 2005

[Nesterov, 2005] Y. Nesterov, Excessive gap technique in nonsmooth convex minimization. *SIAM Journal of Optimization* **16,** 1 (2005), 235–249.

[Nicolai and Hilderman, 2009] G. Nicolai and R.J. Hilderman, No-limit Texas Hold'em poker agents created with evolutionary neural networks. In: Proc. of *IEEE Symposium on Computational Intelligence and Games*, 2009. CIG 2009, 125–131.

[Rubin and Watson, 2009] J. Rubin and I. Watson, A memory-based approach to two-player Texas hold'em. In: *Proc. of the 22nd Australasian Joint Conference on Advances in Artificial Intelligence*, 2009, AI'09, 465–474.

[Rubin and Watson, 2011] Jonathan Rubin and Ian Watson, Computer poker: a review. *Artificial Intelligence*, **175,** 5-6 (Apr 2011), 958–987.

[Salonen, 2011] T. Salonen, The bluffbot website, http://www.bluffbot.com/, checked 26.11.2011

[Schnizlein, 2009] David Paul Schnizlein, State translation in no-limit poker. M. Sc. Thesis, University of Alberta, 2009.

Hiljainen tietämys – historia ja perusteet

Timi Vienola

Tiivistelmä.
Tässä tutkielmassa tarjoan katsauksen hiljainen tietämys -käsitteen historiaan ja luonteeseen, sen yksilö- ja organisaatiotason teorioihin sekä kriittisiä näkökulmia edellisiin aiheisiin. Lisäksi tutkin hiljaisen tietämyksen merkitystä organisaatioiden kilpailukyvyn kannalta.

Avainsanat ja -sanonnat: Tietämys, hiljainen tietämys
CR-luokat: H.1.1, K.6.1

1. Johdanto

Viime vuosikymmenten tietotekninen kehitys on ollut vauhdittamassa kiinnostusta tietoon ja tietämykseen sekä niiden tutkimukseen. Internet on mahdollistanut, että tieto on jokaisen ihmisen ulottuvilla nopeasti ja helposti, mistä tahansa aiheesta. Lisäksi työn muodot ovat muuttuneet. Yhä suurempi osa ihmisistä työskentelee tehtävissä, joissa yksilön omalla ja organisaation hallussa olevalla tiedolla on keskeinen merkitys organisaatioiden menestykseen. Yksilön tietämysresurssit ovat lisäksi tärkeä tekijä menestykseen työntekijämarkkinoilla. Olemme siirtyneet globaalisti kohti tietoyhteiskuntaa, jossa informaatiotekniikan kehittymisen ja yleisen globalisaatiosuuntauksen myötä tiedosta on tullut sekä keskeinen tekijä yhteiskunnassa että kilpailuresurssi organisaatioissa [Kolehmainen, 2001].

Tiedosta ja tietämyksestä puhuttaessa sekoitetaan usein eri käsitteet. Sanaa ”tieto” käytetään usein yleiskäsitteenä kaikille tiedon eri muodoille. Tietoon liittyviä asioita tarkastellessa on kuitenkin tärkeää tehdä selväksi, minkälaisissa muodoissa tieto voi esiintyä. Tieto voidaan luokitella kolmeen eri peruskäsitteeseen: dataan, tietoon ja tietämykseen [Wilson, 2002].

Michael Polanyi esitteli ensimmäisenä hiljainen tietämys -käsitteen vuonna 1966 ilmestyneessä teoksessaan *The Tacit Dimension* [Ambrosini and Bowman, 2001]. Polanyin alkuperäisen käsitteen ympärille on syntynyt ristiriitaisia teorioita, mikä kuvaa hiljaisen tietämyksen vaikeasti hahmotettavaa ja monimuotoista luonnetta. Yleisellä tasolla voi kuvata hiljaisen tietämyksen olevan ihmisen kognitiivisiin rakenteisiin ja käytännön toimintaan juurtunutta tietämystä, jota ei pysty ulospäin eksplisiittisesti suoraan kuvaamaan tai kertomaan. Luonteestaan johtuen sitä pidetään eräänä tärkeimmistä resursseista globaalissa tietotaloudessa toimivien organisaatioiden kilpailukyvyn kannalta. [Nonaka and Takeuchi, 1995]

Tässä tutkielmassa selvitän hiljaisen tietämyksen käsitettä alkaen tiedon eri muodoista. Laajennan aiheen käsittelyä organisaatiotasolle – hiljaisen tietämyksen merkitykseen organisaatioille ja sen hyödyntämiselle organisaatioissa. Lisäksi tutkin useita eri kriittisiä näkemyksiä koskien hiljaisen tiedon määritelmää ja koko käsitteen olemassaoloa. Lopuksi tarkastelen mahdollisuuksia käsitteen empiiriseen tutkimiseen.

2. Tiedon eri muodoista

Puhekielessä käytetään usein sanaa ”tieto” kuvaamaan kaikenlaisia tiedon eri muotoja. Ymmärtääksemme hiljaisen tietämyksen luonnetta, on tärkeää aloittaa aiheen avaaminen eri tietomuotojen peruskäsitteistä. Näitä ovat data, informaatio ja tietämys. Käsitteet muodostavat hierarkkisen ketjun, jossa jokaisen tason käsite rakentuu edeltäjänsä varaan ja laajentaa sen merkitystä [Zins, 2007].

Data eli tietoalkio on tietoon liittyvistä peruskäsitteistä tasoltaan alhaisin. Tietoalkio on kuvaus jostakin mitattavasta reaalimaailman objektin tai ilmiön tilasta. Huomattavaa on, että datalla ei ole kontekstia. [Kangassalo, 2004; Boisot and Canals, 2004]. Se koostuu yksinkertaisista totuuksista, esimerkiksi numeerisen mittausaineiston muodossa. Tällainen numeerinen aineisto voi olla esimerkiksi potilaiden pulssimittaustuloksia. Data-aineiston sisältämät numeeriset alkiot eivät merkitse vastaanottajalle mitään ennen kuin selviää niihin liittyvä konteksti. Data muuttuukin informaatioksi, kun se saadaan liitettyä vastaanottajan kannalta ymmärrettävään asiayhteyteen [Wilson, 2002].

Tietämys-käsitteen filosofisella tutkimisella on pitkä historia. Platonin mukaan tietämys on ”perusteltu tosi uskomus” ja tämä määritelmä on juurtunut myös länsimaiseen epistemologiaan, jossa on perinteisesti tehty selkeä jako objektin ja subjektin, tietäjän ja tiedettävän, välille. Tämän lisäksi länsimaisessa epistemologiassa on kaksi eriävää oppisuuntaa, rationalismi ja empirismi. Rationalismin mukaan tietämys on seurausta deduktiivisesta päättelystä, esimerkiksi matematiikka pohjautuu tällaiseen ajatteluun. Empirismi taasen nojaa ajatukseen, että tietämys pohjaa aistinvaraisista tuntemuksista johdettuihin induktiivisiin päättelyihin. [Nonaka and Takeuchi, 1995, pp. 20–22]

Wilson [2002] väittää, että tietämys on tulkintaan, ymmärtämiseen ja oppimiseen liittyviä ihmisen omia ja jakamattomia mentaalisia prosesseja. Hänen mielestään tietämyksen muodostamiseen liittyy kuitenkin tiivis kanssakäyminen ja informaation jakaminen ulkomaailman kanssa. Nonaka ja Takeuchi [1995] jakavat tämän näkemyksen ja vertailevat tietämyksen ja informaation luonteita kolmesta eri näkökulmasta. He ensinnäkin väittävät, että tietämys ja informaatio syntyvät kumpikin ihmisten välisissä sosiaalisissa kanssakäymisissä ja on tiiviisti yhteydessä kulloiseen tilanteeseen ja asiayhteyteen. Toiseksi,

informaatiosta poiketen tietämys liittyy tiiviisti ihmisen omiin uskomuksiin ja näkökantoihin. Näin tietämys ei edusta absoluuttista totuutta, vaan se on ihmisen henkilökohtainen ”perusteltu uskomus”. Kolmanneksi, ihmisen tietämyksellä ja toiminnalla on vahva yhteys. Informaatio tarjoaa ihmisille tietoa ympäristön tapahtumien ja objektien tulkintaan. Tätä tietoa hyväksi käyttäen ihminen rakentaa tai muokkaa omaa tietämystään.

3. Michael Polanyi – hiljaisen tietämyksen isä

Tiedemies ja filosofi Michael Polanyi on yleisesti tunnustettu hiljainen tietämys -käsitteen keksijäksi, joka esitteli käsitteen ensimmäisen kerran teoksessaan *The Tacit Dimension* (1966) [Tsoukas, 2002; Nonaka and Takeuchi, 1995 p. 38; Ambriosini and Bowman, 2001]. Lukuisat nykyaikaiset tulkinnat hiljaisesta tietämyksestä pohjaavat edelleen Polanyin vuosikymmenten takaisiin johtopäätöksiin. Polanyi tiivisti hiljaisen tietämyksen ydinajatuksen ihmisen kyvyksi tietää enemmän kuin hän pystyy kertomaan [Nonaka and Takeuchi 1995, p. 60].

3.1. Mielen ja kehon saumaton liitto

Länsimaisissa epistemologioissa on yleisesti tehty jako subjektin ja objektin, tietäjän ja tiedettävän, välille. Näin on tehty jakolinja aineettoman mielen ja aineellisen kehon välille. Polanyin päätelmien mukaan ei ole olemassa mitään objektiivista, ihmisestä irrallista tietämystä. Hän väittää tietämiseen liittyvän aina kiinteästi ja erottamattomasti ihmisen taidollista toimintaa ja käyttää tästä esimerkkinä maantieteellisen kartan lukemista. Kartta toimii ainoastaan kuvauksena jostakin tietystä maantieteellisestä alueesta ja sen käyttämiseen tarvitaan aina ihmisen taidollista toimintaa. On ensinnäkin osattava paikallistaa itsensä ja haluttu päämäärä kartalta. Tämän lisäksi on kyettävä suunnistamaan haluttuun päämäärään. Kartta ei kuitenkaan tätä toimintoa suorita, vaan toimii ainoastaan instrumenttina ihmisen tietämykselle tämän tehdessä tulkintaa reaalimaailman ja karttakuvauksen välillä. [Tsoukas, 2002]

Polanyi korostaa useasti kehon keskeistä roolia tietämiseen liittyvässä toiminnassa. Uuden taidon oppiminen vaatii eksplisiittisen tiedon ja ihmisen kehon aistimusten kiinteää vuorovaikutusta. Ihmisellä voi olla jo etukäteen tietoa hänelle uudesta taidosta, mutta keho ja sen aistituntemukset toimivat työkaluina tämän uuden tutkimisessa ja oppimisessa. Polanyi käyttää esimerkkinä polkupyörällä ajamista. Ennen kuin tätä taitoa aletaan opetella, on ihmisellä yleensä tiedossa perusasioita ja sääntöjä pyörällä ajamisesta – pyörää ohjataan ohjaustankoa kääntämällä ja tasapaino säilyy pysymällä liikkeessä polkimia polkemalla. Nämä ohjeet eivät yksinään riitä opettamaan ihmiselle tätä uutta taitoa. Mukaan tarvitaan ihmisen kehollinen toiminta, jolloin omia aistihavaintojaan jo

saatuun tietoon yhdistämällä ihminen alkaa sisäistää polkupyörällä ajamisen taitoa. Opitut taidot ovat hallitsijalleen suurilta osin näkymättömiä, minkä vuoksi niiden purku osatekijöihinsä on mahdotonta. [Tsoukas, 2002]

3.2. Hiljaisen tietämyksen elementit

Polanyin mukaan ihmiset oppivat ja kehittävät taitojaan heuristisesti, virheiden ja onnistumisten kautta etenemällä. Opittua taitoa harjoittavaan toimintaan liittyy kahdenlaisia tietoisuuselementtejä. Tiedostamattomat elementit ovat taidon harjoittamiseen liittyviä osatekijöitä, jotka eivät kuitenkaan ole taidon harjoittajan huomion kohteena. Hänen huomionsa kohde on sen sijaan ensisijaisessa kohteessa – toimintansa päämäärässä. Nämä kaksi tietoisuuteen liittyvää elementtiä ovat yhtä tärkeässä roolissa toiminnan onnistuneen lopputuloksen kannalta. Niillä on kuitenkin toisiinsa nähden eksklusiivinen luonne, sillä huomion kohteen siirtyessä tiedostamattomiin elementteihin tehden niistä näin ensisijaisen huomion kohteen, ne menettävät merkityksensä onnistuneen taidon harjoittamisen kannalta ja toiminta ei ole enää oikeellista. [Tsoukas, 2002]

Edeltävää selventääkseen Polanyi käyttää esimerkkinä naulan lyömistä seinään vasaran avulla. Ihminen on tietoinen vasarasta kädessään, siitä miltä se tuntuu ja miten hän sitä puristaa. Vasara ja siihen liittyvät aistimukset eivät kuitenkaan ole hänen huomionsa kohde, vaan toimivat naulan lyömisessä tiedostamattomana elementtinä. Saadakseen lyötyä naulan onnistuneesti seinään hänen toimintansa ensisijainen huomion kohde on naula ja lyöntinsä siihen. Jos hän kesken toiminnan siirtää huomionsa johonkin tiedostamattomaan elementtiin, esimerkiksi otteeseensa vasarasta, naulan lyöminen seinään epäonnistuu. Ihmisen taidollinen toiminta koostuu vaihtelevasta määrästä tällaisia tiedostamattomia elementtejä, jotka hänen täytyy kuitenkin alitajuisesti ymmärtää, jotta toiminta olisi onnistunutta. Ihminen siis alitajuisesti tunnistaa nämä tiedostamattomat, ”hiljaiset” elementit, mutta ei pysty eksplisiittisesti erittelemään tai nimeämään niitä. Polanyi väittääkin, että hiljainen tietämys muodostaa kolmion koostuen tiedostamattomista elementeistä, ensisijaisesta huomion kohteesta ja ihmisestä, joka toiminnassaan linkittää nämä kaksi elementtiä. Tästä hän johtaa, että kaikenlainen tietämys on aina henkilökohtaista ja tietämiseen liittyy aina toimintaa. [Tsoukas, 2002]

3.3. Hiljainen tietäminen ja uuden tietämyksen sisäistäminen

Tsoukas [2002] selventää Polanyin hiljaisen tietämisen käsitettä käyttäen Polanyin kertomaa esimerkkiä hammaslääkäristä. Tämän mukaan hiljaiseen tietämiseen kuuluu kolme eri aspektia. Toiminnallinen aspekti liittyy hammaslääkärin kykyyn keskittyä tutkimaan työkalullaan eli hammaspiikillä potilaan hampaita, mikä on hänen toimintansa ensisijainen huomion kohde ja tavoite.

Tämä on mahdollista, koska hammaslääkäri tunnistaa alitajuisesti toimintaansa liittyvät tiedostamattomat elementit – hammaspiikin tunteen kädessään ja sen aiheuttamat aistimukset. Havainnollinen aspekti syntyy, kun hammaslääkärin toiminta synnyttää aivan uuden havaittavan aistituntemuksen. Hammaslääkäri ei enää tunnekaan hammaspiikin aistimuksia kädessään, vaan nämä tuntoaistimukset ikään kuin siirtyvät hammaspiikin kärkeen, jonka kosketukset hampaisiin hammaslääkäri voi nyt suoraan aistia. Tämä johtaa hiljaisen tietämisen semanttiseen aspektiin, jossa hammaslääkäri saa aistimustensa kautta tietoa hampaista.

Uuden tietämyksen sisäistäminen on Polanyin mukaan oleellinen käsite hiljaisen tietämyksen ja tietämisen yhteydessä [Nonaka and Takeuchi, 1995 p. 60]. Ihmisen oppiessa uutta hän saa itselleen uudenlaista tietoa ja tietämystä, joka hänen on ensin opittava, sisäistettävä. Polanyin mukaan uusi tietämys muuttuu hiljaiseksi tietämykseksi, kun ihminen on sen sisäistänyt. Hammaspiikin käyttö, kartanlukemisen taito ja vasaralla naulan lyöminen ovat kaikki ihmisen sisäistämää tietämystä, joka toimii Polanyin mukaan työkaluna, ihmisen kehon ja mielen jatkeena. Jotta näitä työkaluja ihminen voi toiminnassaan sujuvasti hyödyntää, on niiden oltava tiedostamattomia, toiminnan mahdollistavia instrumentteja. [Tsoukas, 2002]

Polanyi lisäksi korostaa, että ihmisen tietoisuus toimintaansa liittyvistä elementeistä on dynaamista ja liittyy aina toiminnan kontekstiin. Kun toiminnallinen tilanne muuttuu, niin samalla muuttuu myös ihmisen tietoisuuden taso toimintaansa liittyviä elementtejä kohtaan. Kun ihminen ajaa autoa, hän käyttää vaihteita tiedostamattomasti. Hänen tarvitsee ainoastaan tietää, miten vaihteita vaihdetaan ja miten vaihto vaikuttaa autolla ajamiseen. Kun ihminen on sisäistänyt autolla ajamisen taidon, vaihteiden vaihdosta tulee tiedostamatonta toimintaa. Automekaanikko taasen joutuu omassa toiminnassaan tietämään auton vaihteista huomattavasti laajemmin ja eri näkökulmasta. Automekaanikon toiminnassa vaihteet ovat hänen ensisijainen huomionsa kohde. [Tsoukas, 2002]

4. Nonaka ja Takeuchi – organisaatioiden hiljainen tietämys

Kiinnostus hiljaisen tietämyksen tutkimukseen on ollut tasaisessa kasvussa aina 1990-luvun puolivälistä saakka. Suuri vaikutus tähän on ollut Nonakan ja Takeuchin teoksella *The Knowledege-Creating Company* [Tsoukas, 2002; Wilson, 2002]. Teoksessa hiljainen tietämys käsitetään organisaatioissa sijaitsevana tietämysresurssina, joka on mahdollista levittää koko organisaation käyttöön. Tämän mahdollistaa tietämyksen luonne – Nonaka ja Takeuchi väittävät, että hiljainen tietämys on muunnettavissa eksplisiittiseksi tiedoksi ja toisin päin.

4.1. Teorian lähtökohdat

Japanilaisten ja länsimaisten organisaatioiden suhtautuminen tietoon ja tietämykseen on ollut hyvin erilaista. Tällä oli ollut suuri merkitys japanilaisten organisaatioiden menestykseen 1990–luvun puoliväliin tultaessa. Menestys oli seurausta japanilaisten kyvystä luoda uutta tietämystä, levittää se koko organisaation käyttöön ja lopulta tiivistää tietämys innovatiivisiin tuotteisiin ja palveluihin. Tämä organisaation laajuinen tietämyksen luomis- ja levitysketju selittää japanilaisten organisaatioiden jatkuvaa innovaatiokierrettä, joka taasen loi organisaatiolle kilpailuetua markkinoilla. [Nonaka and Takeuchi, 1995, pp. 3–6]

Länsimaalaiset organisaatiot käsittivät tietämyksen lähinnä sen eksplisiittisessä muodossa. Eksplisiittinen tietämys tarkoittaa tietämystä, joka on ilmaistu formaalilla ja systemaattisella tavalla esimerkiksi kaavojen tai tekstikirjojen muodossa. Eksplisiittinen tietämys on täten helposti kommunikoitavissa ja levitettävissä eteenpäin. Japanilaiset organisaatiot taasen ymmärsivät eksplisiittisen tiedon kattavan vain osan olemassa olevasta tietämyksestä. Heidän käsityksensä mukaan tietämys on suurelta osin henkilökohtaista ja vaikeasti ilmaistavissa – hiljaista tietämystä. Tietämyksen ymmärtäminen eksplisiittisenä tai hiljaisena tietämyksenä loi jakolinjan länsimaisten ja japanilaisten organisaatioiden tapaan lähestyä tietämyksen käsitettä. [Nonaka and Takeuchi, 1995, pp. 8–9]

Nonaka ja Takeuchi väittävät hiljaisen tietämyksen olevan kahtalainen luonteeltaan. Sillä on tekninen ulottuvuus, joka liittyy ihmisen toimintaan sisältyvään tietotaitoon. Ihmiset omaavat moninaisia pitkälle kehittyneitä taitoja, joita on kuitenkin vaikea ulkopuolisille selittää tai opettaa. Esimerkiksi japanilainen baseball-pelaaja ei osannut selittää, miksi hän on niin hyvä lyömään palloa, ja hänen mukaansa lyönti ”pitää vain tuntea”. Teknisen ulottuvuuden lisäksi hiljainen tietämys sisältää kognitiivisen dimension, joka koostuu ihmisen uskomuksista, arvoista ja mentaalimalleista, joiden kautta ihminen koostaa käsityksensä reaalimaailmasta. [Nonaka and Takeuchi, 1995, pp. 8–9]

4.2. Tietämyksen muunto

Nonakan ja Takeuchin teorian kulmakivenä toimii ajatus, jonka mukaan tietämys on joko eksplisiittistä tai hiljaista ja se on mahdollista muuntaa muodosta toiseen. Tämä mahdollistaa tietämyksen mobilisoinnin, uuden tietämyksen luonnin ja tietämyksen levittämisen koko organisaation käyttöön. Erilaisia muuntomuotoja on neljä: hiljaisen tietämyksen muunto hiljaiseksi, hiljaisen tietämyksen muunto eksplisiittiseksi, eksplisiittisen tietämyksen muunto eksplisiittiseksi ja eksplisiittisen tietämyksen muunto hiljaiseksi tietämykseksi (ks. kuva 1).

	Hiljaiseksi Tietämykseksi	Eksplisiittiseksi Tietämykseksi
Hiljaisesta Tietämyksestä	Sosiaalistaminen	Ulkoistaminen
Eksplisiittisestä Tietämyksestä	Sisäistäminen	Yhdistäminen

Kuva 1: Tietämyksen neljä erilaista muuntomuotoa [Nonaka and Takeuchi, 1995, p. 62]

Hiljaista tietämystä on mahdollista siirtää vastaanottajan hiljaiseksi tietämykseksi ”sosiaalistamisen” kautta. Tämä tapahtuu ihmisten sanattomassa kanssakäymisessä, oppilas–opettaja-luonteisessa sosiaalisessa kontekstissa. Oppilas tarkkailee opettajansa toimintaa ja pyrkii sisäistämään opeteltavaa taitoa hänen toimintaansa matkimalla. Tärkeää tällaisessa tilanteessa on yhteisesti jaettu kokemus, joka koostuu jaetusta tunnetilasta ja asiayhteydestä. Selventäväksi esimerkiksi voi ottaa Ikuko Tanaken, joka pyrki oppimaan leipätaikinan vaivaamiseen liittyviä saloja 1980-luvun lopulla. Hän toimi projektissa, jonka päämääränä oli tuoda markkinoille automaattinen leipäkone. Ongelmaksi muodostui taikina, josta leipä leivottiin – siitä ei saatu kunnollista. Tanake päätti pestautua paikalliselle leipurimestarille harjoittelijaksi. Leipurin toimintaa seuraamalla ja imitoimalla hän ajan myötä sisäisti taikinan leipomiseen liittyvän vaivaamistekniikan, joka oli laadukkaan leivän salaisuus. [Nonaka and Takeuchi, 1995, pp. 62–64]

Hiljaisen tietämyksen muuntaminen eksplisiittiseksi tiedoksi tapahtuu ”ulkoistamalla”, muuntamalla yksilön hiljaista tietämystä kirjoitettuun tai muuhun siirrettävään muotoon. Uuden tietämyksen luonti organisaatioissa nojaa vahvasti juuri tähän muuntomuotoon, jossa yksilön hiljaista tietämystä pyritään ulkoistamaan käsitteiksi. Hiljaisen tietämyksen luonteesta johtuen sen muuntaminen suoraan yksiselitteiseksi tiedoksi on mahdotonta. Muuntaminen

tapahtuukin käyttäen avuksi kielikuvia ja analogioita, joiden kautta hiljaista tietämystä pyritään kuvaamaan ulospäin. Kielikuvien eli metaforien avulla pyritään ulkoistamaan hiljaiseen tietämykseen liittyviä yksilön mentaalisia käsitteitä ja mielikuvia. Ilmaistuihin metaforiin voi sisältyä suuriakin ristiriitaisuuksia. Analogioiden avulla näistä ristiriitaisuuksista pyritään löytämään yhteneväisyyksiä ja harmonisoimaan kielikuvista yksilön sisäistä tietämystä kuvaavia eksplisiittisiä käsitteitä. Ikuko Tanake ulkoisti taikinan vaivaamiseen liittyvää hiljaista tietämystään kuvaamalla vaivaamisliikkeen olevan ”vääntävää venyttämistä” (”twisting stretch”). Tämän uuden käsitteen avulla insinöörit osasivat tehdä tarvittavat muutokset leipäkoneeseen, jonka avulla taikinasta saatiin oikeanlaista. [Nonaka and Takeuchi, 1995, pp. 64–67]

”Yhdistäminen” on tietämyksen muunnos, jossa eksplisiittisestä tietämyksestä luodaan uutta eksplisiittistä tietoa. Organisaatiossa eksplisiittistä tietoa vaihdetaan ja vastaanotetaan esimerkiksi kirjoitettujen dokumenttien kautta. Kun tätä vastaanotettua tietoa lajitellaan, yhdistellään vanhaan tietoon ja muokataan, yksilö muodostaa itselleen uutta eksplisiittistä tietämystä. Tätä muutosmuotoa tapahtuu jatkuvasti esimerkiksi opiskellessa. Opiskelija saa eksplisiittistä tietoa suullisesti opettajilta tai tekstimuodossa kirjoista. Tätä saatua tietoa yhdistellessään ja järjestäessään hän luo tiedon pohjalta uusia johtopäätöksiä ja luo näin itse uutta eksplisiittistä tietämystä. [Nonaka and Takeuchi, 1995, pp. 67–68]

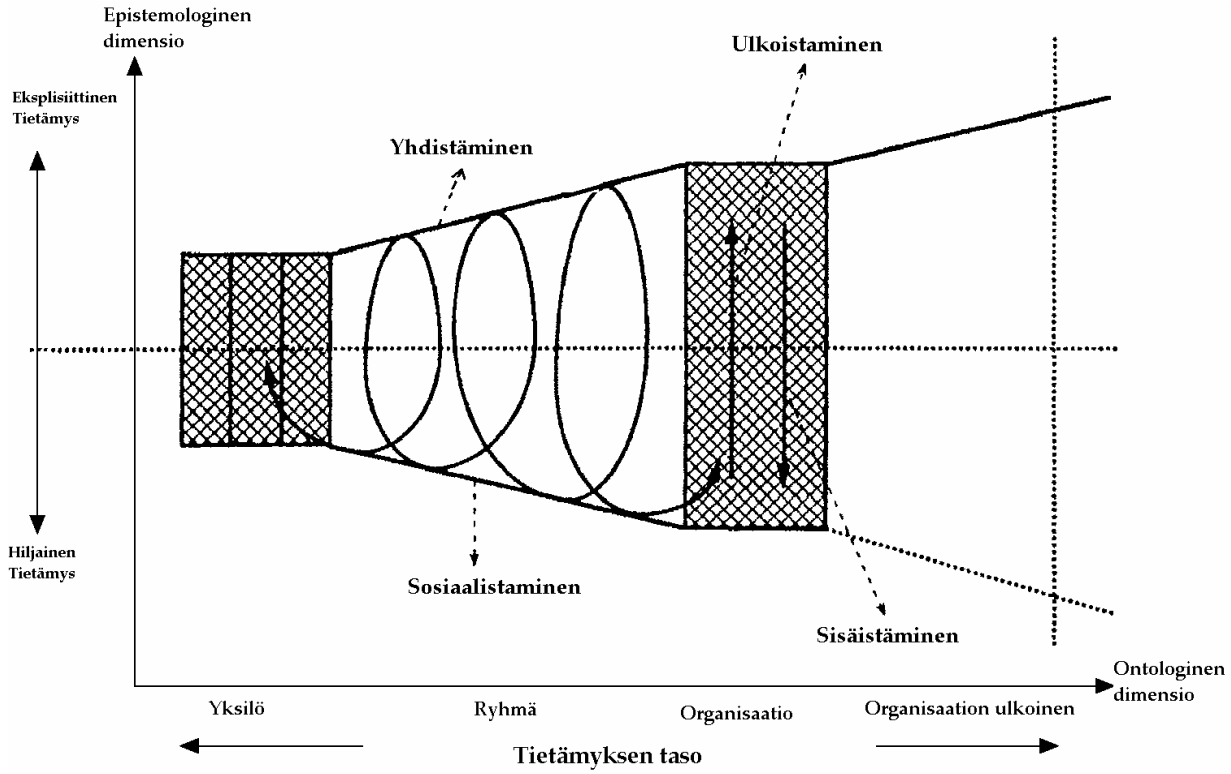
Tietämyksen muuntaminen eksplisiittisestä jälleen hiljaiseksi tietämykseksi liittyy tiiviisti ihmisen oppimiseen oman toimintansa ja kokemustensa kautta. ”Sisäistäminen” tapahtuu ihmisen oppiessa jotakin uutta eksplisiittisen tietämyksen kautta ja opitun sulautuessa osaksi ihmisen mentaalisia ja kognitiivisia rakenteita. Ihminen voi sisäistää käsitteitä tai taitoja myös henkilökohtaisen kokemisen kautta. Esimerkiksi japanilaisessa Matsushitan organisaatiossa vuonna 1993 pyrittiin vähentämään työntekijöiden vuosittainen työmäärä 1800 tuntiin. Tavoitteena oli työntekijöiden innovatiivisuuden ja luovuuden kasvattaminen toimenpiteen avulla. Eksplisiittinen käsite – 1800 vuosittaista työtuntia – sisäistettiin työntekijöihin kuukauden mittaisella koejaksolla, jossa kokonaistyöaika oli 150 tuntia. Tämän kokemuksen avulla työntekijät sisäistivät, miltä 1800 vuosittaisen työtunnin käsite merkitsi. [Nonaka and Takeuchi, 2005, pp. 69–70]

4.3. Tietämyksen luomiskierre ja sen mahdollistavat tekijät

Organisaatioissa tapahtuva uuden tietämyksen luominen vaatii jatkuvaa tietämyksen muunnosta olomuodosta toiseen. Tätä spiraalimaisesti eri muunnosmuotoja läpikäyvää prosessia Nonaka ja Takeuchi kutsuvat nimellä ”tietämyskierre”, jossa jokainen uusi kierros rakentuu inkrementaalisesti edeltävän pääl-

le. Kuten edellisessä kappaleessa esitettiin, jokaisen muunnoksen synnyttämä tietämys eroaa edeltäjästään, samalla rakentuen sen varaan. Sisäistämisen kautta syntynyt uusi hiljainen tietämys tarjoaa resursseja jälleen uutta kierrosta varten. [Nonaka and Takeuchi, 1995, pp. 70–72]

Kierre ei pyöri ainoastaan tietämyksen eri muotojen välillä. Sen voidaan nähdä laajenevan samanaikaisesti myös toisella akselilla, organisaation rakennetasolla (ks. kuva 2). Nonaka ja Takeuchi korostavat voimakkaasti, että kaikki tietämys on lähtöisin ihmisistä. Organisaatio ei omana entiteettinään luo tietämystä, vaan ihmiset ovat kaiken tietämyksen synnyn takana. Organisaation laajuisen tietämyksen luonti alkaa aina yksittäisestä ihmisestä. Ihmisten välisen kanssakäymisen seurauksena tietämys muuttaa muotoaan, lisääntyy ja leviää läpi organisaatiorakenteiden – yksilö-, ryhmä- ja organisaatiotasolta aina ulkopuolisiin organisaatioihin saakka. [Nonaka and Takeuchi, 1995, pp. 72–73]



Kuva 2: "Tietämyskierre" – tietämyksen leviäminen ja vahvistuminen kahdella eri tasolla [Nonaka and Takeuchi, 1995, p. 62].

Nonakan ja Takeuchin [1995, pp. 74–83] mukaan organisaation on täytettävä viisi erilaista ehtoa, jotka mahdollistavat uuden tietämyksen synnyn ja leviämisen organisaatiossa. Ensinnäkin organisaatiolla on oltava intentio – tavoite ja halu – saavuttaa omat päämääränsä. Käytännössä tämä näkyy haluna kehittää organisaation strategiaa, joka antaa raamit sille, minkälaista tietämystä organisaatiossa halutaan luoda ja käyttää. Intentio muodostaa täten laatukritee-

rit, joita vastaan uutta tietoa ja tietämystä verrataan. Intentio peilaa siis organisaation arvoja.

Toiseksi, organisaatiossa on vaalittava yksilön autonomiaa ja organisaatiorakenteiden joustavuutta. Tällaisessa toimintaympäristössä yksilöllä on vapaus toimia luovasti ja asettaa itse tavoitteita toiminnalleen organisaation intention pohjalta. Itsenäinen työskentely motivoi uuden tietämyksen luomiseen. Kun toimintaympäristö ei ole tarkasti säädelty, se luo myös kasvualustan sattumanvaraisten tilanteiden ja mahdollisuuksien syntymiselle. Autonomiaa tukee myös tiimityöskentely. Poikkitoiminnalliset tiimit koostuvat useiden eri tehtävien osaajista, joiden välinen kanssakäynti luo pohjaa uudenlaisen tietämyksen synnylle.

Kolmantena ehtona Nonaka ja Takeuchi esittävät, että organisaation vakiintuneita rakenteita ja työtapoja ravistellaan tasaisin väliajoin sekä aiheutetaan toimintaan luovaa kaaosta. On luonnollista, että ajan mittaan organisaation rakenteet ja ihmisten työtavat vakiintuvat luoden rutiinia toimintaan. Kun näitä vakiintuneita tapoja rikotaan, joutuvat ihmiset tarkastelemaan toimintaansa ja näkökantojaan uudelta kantilta. Tämä johtaa yleensä kommunikoinnin lisääntymiseen ympäristön kanssa ja tämä kanssakäyminen mahdollistaa uuden tietämyksen syntymistä. Organisaation toimintaan luo kaaosta yleensä ulkoinen toimintaympäristö, esimerkiksi vahvistuneen kilpailijan tai markkinoiden muun kriittisen muutoksen myötä. On hyödyllistä luoda hallittua kaaosta organisaation toimintaan myös sisältäpäin. Tämä synnyttää intensiivisemmän työilmapiirin, joka auttaa löytämään innovatiivisia ratkaisuja ongelmiin ja näkemään uusia mahdollisuuksia. Jotta tällainen keinotekoinen kaaos olisi onnistunutta ja mahdollistaisi uudenlaisen tietämyksen syntymisen, on yksilöillä oltava aikaa oman toimintansa peilaamiseen ja kokemuksistaan oppimiseen.

Tarpeettoman tai päällekkäisen tiedon jakaminen organisaatiossa on neljäs ehto tietämyksen luomiskierteen toiminnalle. Uusien käsitteiden ja tiedon jakaminen organisaation sellaisillekin jäsenille, jotka eivät niitä suoranaisesti toiminnassaan tarvitse, vaikuttaa organisaatioon kolmella eri tavalla. Tarpeettoman tiedon mukana leviää myös hiljaista tietämystä, koska vastaanotetun tiedon avulla voidaan aistia muualla organisaatiossa työskentelevien käsityksiä ja näkökulmia. Lisäksi organisaationlaajuisesti leviävä tieto tasapäistää eri tasoilla työskenteleviä ja rikkoo hierarkkisia rakenteita mahdollistaen samalla uusien kommunikointikanavien synnyn. Päällekkäinen tieto antaa ihmisille myös käsityksen heidän omasta asemastaan organisaation kokonaistoiminnassa ja samalla ohjaa heidän omaa toimintaansa. Tiedon levittämistä organisaatiossa voidaan edistää esimerkiksi toimintojen välisellä työntekijävaihdolla.

Nonakan ja Takeuchin viimeisen ehdon mukaan organisaatiossa on oltava tarvittava määrä tiedollista ja taidollista moninaisuutta. Menestyksekkään toiminnan edellytys on, että organisaation sisäinen monimuotoisuus vastaa toimintaympäristön moninaisuutta, jotta ympäristön tarjoamiin haasteisiin ja ongelmiin voidaan vastata organisaation toiminnassa. Tämän takia on varmistettava, että organisaation kaikilla jäsenillä on yhtäläinen mahdollisuus hyödyntää organisaation koko tietämysresurssia mahdollisimman vaivattomasti. Näin varmistetaan, että organisaatioon ei synny tietämyspimentoja, jotka eivät pysty vastaamaan tehokkaasti ja nopeasti ympäristönsä muutoksiin. Myös organisaatiorakenteen pitäminen matalana ja sen eri toiminnot tiiviisti linkitettynä mahdollistaa tehokkaan tietämyksen vaihdon ja käytön. Samalla organisaatio pysyy joustavana toiminnassaan.

4.4. Viisi vaihetta tietämyksen luomiseen organisaatioissa

Edellä on esitelty tietämyksen neljä erilaista muunnosmuotoa ja viisi ehtoa, jotka edistävät organisaatiossa tapahtuvaa tietämyksen luomista. Näiden lisäksi Nonaka ja Takeuchi [1995, pp. 83–89] esittävät teoriassaan viisivaiheisen mallin tietämyksen luomisprosessista organisaatiossa.

Prosessin ensimmäisenä vaiheena on yksilöiden hiljaisen tietämyksen jakaminen työtiiminsä kesken. Nonaka ja Takeuchi korostavat useasti kaiken uuden tietämyksen alkulähteen olevan yksilöissä, heidän hiljaisessa tietämyksessään. Ihmisten keskustellessa ja vaihtaessa kokemuksiaan luottamuksellisessa ilmapiirissä, he samalla jakavat hiljaista tietämystään. Tämä liittyy tiiviisti edellä esitettyjen muutosmuotojen sosiaalistamisvaiheeseen. Yksilön hiljaisesta tietämyksestä tulee koko ryhmän yhteistä hiljaista tietämystä. Samalla tietämys voimistuu, kun se on tulos kaikkien jäsenten omista arvoista, taidoista ja henkilökohtaisesta tietämyksestä. Nonaka ja Konno [1998] esittelevät käsitteen ”Ba”, jonka he liittävät tiiviisti tähän prosessin ensimmäiseen vaiheeseen. ”Ba”-käsite merkitsee fyysistä, virtuaalista tai mentaalista tilaa, missä ihmiset voivat jakaa hiljaista tietämystään.

Prosessin toinen vaihe liittyy hiljaisen tietämyksen ulkoistamisvaiheeseen. Ryhmällä on jaettu hiljainen tietämys, jonka he nyt pyrkivät ulkoistamaan eksplisiittisiksi käsitteiksi. Tämä tapahtuu tiimin sisällä tapahtuvan jatkuvan vuoropuhelun ja mielipiteiden vaihdon avulla. Eksplisiittisten käsitteiden synty ei ole suoraviivaista vaan jatkuva iteratiivinen prosessi.

Kolmannessa vaiheessa tiimin luomat uudet käsitteet perustellaan ja evaluoidaan. Evaluoinnissa arvioidaan uusien käsitteiden arvoa ja hyödyllisyyttä organisaation toiminnan kannalta. Lisäksi arvioidaan, ovatko ne organisaation intention mukaisia. Arviointikriteerit voivat olla muodoltaan hyvin konkreettisia – kvantitatiivisia tai kvalitatiivisia arviointiasteikkoja. Projekteissa syntynei-

tä käsitteitä evaluoidaan myös käyttämällä kriteereinä käsitteitä tai visiota, jotka ylempi johto on antanut projektin toimintaa suuntaamaan. Arvioinnin avulla organisaatio päättää, onko syntynyt uusi käsite todella eteenpäin viemisen arvoinen.

Hyväksytyistä käsitteistä luodaan konkreettisia ilmentymiä prosessin neljännessä vaiheessa. Nonakan ja Takeuchin mukaan tämä tapahtuu luomalla prototyyppejä uusista tuoteinnovaatioista tai toimivia malleja aineettomista innovaatioista, kuten uusista toimintaprosesseista tai organisaation rakennerat-kaisuista. Tämä tapahtuu kombinoimalla organisaatiossa jo olevaa eksplisiittistä tietämystä yhteen tiimin luoman uuden eksplisiittisen käsitteistön kanssa.

Tietämyksen luomisen viidennessä ja viimeisessä vaiheessa koko prosessin aikana syntynyttä uutta tietämystä levitetään organisaatioon sekä sen ulkopuolelle. Organisaation sisällä projekteista saatuja kokemuksia voidaan hyödyntää eri toiminnoissa tai jokin tuoteinnovaatio voi synnyttää samaa konseptia käyttävän kokonaisen tuoteperheen. Nonaka ja Takeuchi korostavat organisaatioiden toimivan avoimessa ympäristössä ja uusi tietämys voi täten levitä myös organisaation ulkopuolisille sidosryhmille. Asiakkaat ja kilpailijat ovat kiinnostuneita uusista tuoteinnovaatioista. Aineettomia innovaatioita voidaan jakaa esimerkiksi organisaation tuotantoketjun ulkopuolisille jäsenille näiden toimintaa tehostamaan tai toimintamalleja yhdenmukaistamaan.

5. Hiljainen tietämys ryhmätasolla

Suuri osa hiljaisen tietämyksen tutkimuksesta on tehty yksilötason tietämystä tutkien. Nonakan ja Takeuchin [1995] mukaan kaikki tietämys on lähtöisin nimenomaan yksilöistä ja heidän omasta hiljaisesta tietämyksestään. He kuitenkin korostavat sosiaalistamisen kautta tapahtuvan hiljaisen tietämyksen jakamisen olevan tärkein yksittäinen vaihe organisaatiossa tapahtuvan uudenlaisen tietämyksen luomisen kannalta. On kuitenkin näkemyksiä, joiden mukaan yksilö- ja ryhmätason hiljainen tietämys eivät ole sama asia tai edes verrattavissa toisiinsa. Ryhmätason hiljainen tietämys nähdään omana kiinteänä yksikkönään, joka ei ole suoraan osatekijöidensä, yksilöiden hiljaisen tietämyksen, summa. [Cook and Brown, 1999; Erden et al., 2008]

Ryhmäksi määritellään joukkoa ihmisiä, jotka toimivat keskinäisessä kanssakäymisessä jonkin yhteisen päämäärän tai aatteen eteen. Ryhmän hiljainen tietämys onkin sosiaalisesti rakentunutta ja syntyy ihmisten kanssakäymisessä, heidän jakaessa yhteisen käsityksen tilanteesta ja halutusta toiminnasta. Täten ryhmän hiljainen tietämys liittyy tiiviisti ryhmän yhteiseen toimintaan. Ryhmä toimii saumattomasti yhteen, kuin jakaen yhteisen tietämyksen. Tällainen tietämys ei täten ole yksilön omistamaa, vaan mahdollista saavuttaa vain ryhmäs-

sä. [Erden et al., 2008]. Esimerkiksi jalkapallojoukkueen pelaajien voidaan nähdä omaavan jaettu hiljainen tietämys. Varsinkin pitkään yhdessä pelanneen joukkueen jäsenet osaavat ikään kuin lukea toistensa ajatuksia yhteispelin ollessa saumatonta.

Ryhmän hiljaisen tietämyksen taso on riippuvainen ryhmän jäsenten monimuotoisuudesta. Tämän monimuotoisuuden yhdistyessä syntyy synergiaa, jolloin ryhmän hiljainen tietämys on suurempaa kuin osiensa, yksilöiden tietämyksen, summa. Tämä selittyy yksilöiden erilaisuudella – toisen yksilön heikkoudet kompensoituvat toisen vahvuuksilla. Jaettu hiljainen tietämys ei kuitenkaan synny hetkessä, vaan sen syntyminen ja syveneminen vaatii ryhmän yhteistä toimintaa ja jaettuja kokemuksia. Jaetut kokemukset ovat hyödyllisiä ainoastaan silloin, kun ryhmä käsittää ne kollektiivisen näkökulman kautta. Tätä kautta syntynyt yhteinen hiljainen tietämys tiivistyy ryhmän arvoihin, normeihin ja kulttuuriin. [Erden et al., 2008]

Edellä mainittujen lisäksi Erden ja muut [2008] löytävät ryhmätason hiljaisesta tietämyksestä vielä kaksi erityispiirrettä. Ensimmäiseksi, jaettu hiljainen tietämys mahdollistaa ryhmän löytää keinot toimia tavalla, joka on ryhmän ”yhteisen hyvän” kannalta paras vaihtoehto. ”Yhteinen hyvä” on riippuvainen tilanteesta ja ryhmän määrittelyssäkin mainitusta yhteisestä aatteesta tai päämäärästä. Toiseksi, ryhmässä syntynyt hiljainen tietämys vähentää ryhmän toimintaan liittyvää epävarmuutta. Ryhmän jäsen kykenee aistimaan, miten muut jäsenet erilaisissa tilanteissa todennäköisesti tulevat toimimaan. Tämän johdosta hän pystyy mukauttamaan omaa toimintaansa muiden jäsenten mukaan. Tämä luo harmoniaa ryhmän toimintaan. Vertailukohdaksi voidaan ottaa ryhmä, jonka jäsenet eivät tunne toisiaan hyvin, eikä jaettua hiljaista tietämystä ole vielä syntynyt. Tällaisen ryhmän jäsenet kokevat epävarmuutta, koska eivät osaa ennakoida toistensa toimintaa ja tämän johdosta ryhmän toiminta on kokonaisuudessaan koordinoimatonta.

Cook ja Brown [1999] korostavat voimakkaasti yksilön ja ryhmän hiljaisen tietämyksen olevan erillisiä entiteettejään, joista kummastakaan ei voi johtaa suoraan toista. Ryhmätason hiljaista tietämystä ei voi jakaa osiin yksilötason tietämykseksi, eikä ryhmätason tietämys ole suoraan yksilötason tietämyksen summa. Ryhmätason hiljaista tietämystä voi syntyä vain ryhmän kanssakäymisessä ja se ohjaa ainoastaan ryhmän kollektiivista toimintaa, joten sillä ei ole merkitystä yksilötasolla. Yksilö voi tietää jostakin aihealueesta osan, mutta koko aihealuetta kattava tietämys muodostuu ryhmässä.

6. Hiljaisen tietämyksen vaikutus organisaation kilpailukykyyn

Nykyajan tietoyhteiskunnassa organisaatioiden hallitsemalla tiedolla ja tietämyksellä on niiden toimintaan ja kilpailukykyyn merkittävä vaikutus [Nonaka and Takeuchi, 1995; Argote and Ingram, 2000; Ambrosini and Bowman, 2001]. Tietämyksestä on tullut resurssi, jota organisaatiot yrittävät haalia itselleen ja varjella kilpailijoiltaan. Tässä luvussa selvitän lyhyesti, miten hiljainen tietämys ja tietämys yleisellä tasolla toimivat organisaation kilpailuresurssina ja mihin niiden tärkeys perustuu.

Resurssipohjainen teoria pyrkii selittämään organisaation kilpailukykyä tarkastelemalla sen resursseja ja resurssien ominaisuuksia. Teorian mukaan resurssit, jotka ovat yhtä aikaa arvokkaita eli pystyvät tuottamaan organisaatiolle arvoa, harvinaisia markkinoilla, vaikeasti kopioitavissa ja vaikeasti korvattavissa, ovat organisaation pysyvän kilpailuedun kannalta elintärkeitä. Hiljainen tietämys on luonteeltaan yksilöihin tai ryhmiin sidottua, jota on vaikea tai jopa mahdoton kuvata eksplisiittisesti ulospäin. Tästä johtuen kilpailijoiden on sitä hyvin vaikea kopioida tai saada muuten itselleen. Ryhmätason hiljainen tietämys syntyy ryhmän kanssakäymisessä ja yhteisessä toiminnassa uniikkina entiteettinä, jota on mahdoton siirtää organisaatiosta ulos siirtämättä koko ryhmää. Lisäksi organisaatioissa voi olla paljon hiljaista tietämystä, joka on niiden toiminnan kannalta tärkeää, mutta jonka olemassaoloa ei organisaatiossa ole vielä havaittu. [Ambrosini and Bowman, 2001]

Argoten ja Ingramin [2000] mukaan organisaatioiden on mahdollista hankkia kilpailuresursseja kahta kautta: ulkoapäin hankkimalla tai sisäisesti organisaatiossa kehittämällä. Jotta ulkoapäin hankinta on kannattavaa, on resurssin tuottaman arvon organisaation toiminnassa katettava sen hankintakustannukset. Argote ja Ingram väittävät tämän puoltavan organisaation sisäistä resurssien kehittämistä ja asettavan täten tietämyksen kilpailuresurssina olennaiseen asemaan. Uuden tietämyksen kehittäminen, säilyttäminen ja levittäminen koko organisaation käyttöön on heidän mukaansa avainasemassa tietämyksen hyödyntämiseen kilpailuresurssina.

Nonaka ja Takeuchi [1995] selittivät teoriassaan, miten organisaatiossa sijaitsevasta hiljaisesta tietämyksestä saadaan luotua aivan uutta tietämystä muuntelemalla sitä olomuodosta toiseen. Heidän mukaansa tällainen tietämyksen luonti on jatkuva dynaaminen prosessi, jonka avulla organisaation tietämystä saadaan kasvatettua ja levitettyä eri tasoille. Argote ja Ingram [2000] näkevät tietämyksen sijaitsevan organisaatiossa kolmessa eri muodossa: ihmisissä, työvälineissä ja teknologioissa sekä työtehtävissä. Lisäksi tietämys esiintyy näiden kolmen perusmuodon erilaisissa yhdistelmissä.

Argote ja Ingram [2000] väittävät organisaatioiden kilpailukyvyn perustuvan tietämyksen luonnin ja säilömisen lisäksi ennen kaikkea niiden kykyyn levittää tietämystä tehokkaasti eri puolille organisaatiota. Tämä voi heidän mukaansa tapahtua ensinnäkin siirtämällä ihmisiä, työvälineitä ja työtehtäviä tai näiden kombinaatioita eri puolille organisaatiota. Tällainen eksplisiittisen tai hiljaisen tietämyksen levittäminen ei ole kuitenkaan ongelmatonta. Siirretty tietämys ei ole välttämättä toimivaa uudessa ympäristössä. Lisäksi erilaisten tietämyselementtien kombinaatiot voivat olla hyvinkin monimutkaisia ja vaikeasti hahmotettavissa. Tämä vaikeuttaa niiden siirtoa, mutta toisaalta tekee niistä kilpailuresursseina äärimmäisen vaikeita kopioida tai siirtää ulos organisaatiosta. Argoten ja Ingramin mukaan tietämystä voidaan levittää myös muokkaamalla ihmisissä, työvälineissä ja työtehtävissä sijaitsevaa tietämystä. Tämä tapahtuu pääasiallisesti koulutuksen ja muunlaisen ihmisten kanssakäymisen välityksellä.

7. Kriittisiä näkemyksiä vallitseviin teorioihin

Michael Polanyi, hiljainen tietämys -käsitteen keksijä, väitti hiljaisen tietämyksen olevan henkilökohtaista, liittyen erottamattomasti toimintaan ja olevan ainakin osaltaan näkymätöntä. Se muodostuu hänen mukaansa kolmesta osasta: tiedostamattomista eli hiljaisista elementeistä, toimintaan liittyvästä huomion keskipisteestä ja ihmisestä näiden kahden linkittäjänä. Hän väittää ihmisen kykenevän taidolliseen toimintaan keskittymällä toimintansa olennaiseen päämäärään ja samalla luottaen alitajuisesti hiljaisiin elementteihin, joista tämä toiminta rakentuu. Nonakan ja Takeuchin [1995] käsityksiä on referoitu laajasti nykyajan tietokirjallisuudessa niin tietämykseen kuin johtamiseen liittyvissä julkaisuissa [Tsoukas, 2002]. He käyttivät mallinsa pohjana Polanyin teoriaa hiljaisesta tietämyksestä. Kritiikkiä on kuitenkin herättänyt Nonakan ja Takeuchin tulkinta Polanyin teoriasta ja näkemyksensä tietämyksen muunneltavasta luonteesta.

Cook ja Brown [1999] väittävät, että hiljainen ja eksplisiittinen tietämys ovat täysin eroavia tietämyksen muotoja, joita ei voi muuntaa toiseksi. Samanlaisen eron he väittävät olevan yksilön ja ryhmän omistaman hiljaisen tai eksplisiittisen tiedon välillä. Nämä neljä tietämyksen muotoa ovat heidän mukaansa kuitenkin keskenään tasa-arvoisia ja ihminen käyttää niistä jokaista omassa toiminnassaan tarpeen mukaan. Cook ja Brown kuitenkin myöntävät, että vaikka tietämys ei voi muuttaa muotoaan, on mahdollista hankkia tietämystä sen toisenlaista muotoa apuna käyttäen. Todelliseen taidon oppimiseen ei kuitenkaan pelkkä etukäteen hankittu tietämys riitä, vaan taidon sisäistäminen tapahtuu aina toiminnan yhteydessä.

Nonaka ja Takeuchi [1995] antoivat esimerkin, jossa Ikuko Tanake sisäisti hiljaiseksi tietämykseksi leipätaikinan vaivaamistekniikan matkimalla ja harjoittelemalla leipurin toimintaa. Myöhemmin Tanake siirsi tämän hiljaisen tietämyksensä eksplisiittiseksi käsitteeksi, jota apuna käyttäen insinöörit suunnittelivat leipäkoneeseen toimivan taikinan vaivaamistoiminnon. Cook ja Brown [1999] väittävät, että kyseisessä esimerkissä ei minkäänlaista tietämyksen muunnosta tapahtunut. Heidän mukaansa kyse oli tuotantotiimin ja sen jäsenten yritys-erehdys -perustaisesta toiminnasta, jossa eri tietämysmuotoja tarpeen mukaan apuna käyttäen ryhmä onnistui luomaan uutta tietämystä – Tanaken kehittämää käsitettä vastaavan vaivaamisliikkeen leipäkoneeseen.

Wilson [2002] on myös kriittinen Nonakan ja Takeuchin näkemyksiä kohtaan ja väittää heidän tulkinneen väärin Polanyin teoriaa. Wilsonin mukaan Polanyin teoria on yksiselitteinen hiljaisen tietämyksen luonteen suhteen – se on piilossa olevaa tietämystä, jota ei voi muuntaa eksplisiittiseksi tai muutenkaan saada esiin. Wilsonin mukaan leipätaikinaesimerkin vaivaamistekniikka ei ollut hiljaista tietämystä, vaan eksplisiittistä tietämystä, joka olisi voitu saada esiin kysymällä leipurilta oikeita kysymyksiä. Wilsonin mukaan tietämys ei siis voi olla hiljaista tietämystä, jos se on mahdollista ilmaista eksplisiittisesti.

Tsoukas [2002] väittää, että Nonakan ja Takeuchin teoria ei huomioi Polanyin hiljaisen tietämyksen käsitteeseen olennaisesti liittyvää tavoittamattomuuden näkökulmaa. Jotta leipäkone-esimerkin Tanake olisi voinut muuntaa vaivaamistaitonsa eksplisiittiseksi tietämykseksi, olisi hänen tunnettava toimintaansa liittyvät, Polanyin teorian mukaiset, hiljaiset elementit. Tämä taasen vaatisi näiden elementtien huomioimista niihin tietoisesti keskittymällä. Tanaken toiminta ei tällöin olisi enää taikinan vaivaamista, vaan taikinan vaivaamistekniikan miettimistä, jonka hiljaiset elementit ovat taasen täysin omanlaisiaan. Tsoukas väittää, että Polanyin teorian mukaan toimintaan liittyvien hiljaisten elementtien ja huomion kohteen keskinäinen suhde on stabiili ja erottamaton. Tästä hän päättelee, että taikinan vaivaamistaitoon liittyviä hiljaisia ja tiedostamattomia elementtejä ei voi koskaan saada esiin ja ilmaistuksi. Tsoukas väittää, että Tanake ei voinut ilmaista eksplisiittisesti koko vaivaamistaitoonsa liittyvää hiljaista tietämystä. Hän ilmaisi ”vääntävä venyttäminen” -käsitteen avulla ainoastaan toimintaansa liittyvää teknistä tietoutta, joka oli mahdollista ulkoistaa säännöiksi ja ohjeiksi.

8. Hiljaisen tietämyksen empiirinen tutkimus

Hiljaisen tietämyksen empiirinen tutkimus ei voi olla suoraviivaista hiljaiseen tietämykseen liittyvien näkymättömyyden ja henkilökohtaisuuden aspektien vuoksi. Kuten edellisessä luvussa on kuvattu, monet haluavat ymmärtää Po-

lanyin tarkoittavan teoriallaan, että hiljaisen tietämyksen luonteeseen liittyy täysin näkymätön osa, joka on täten mahdotonta pyydystää. Jos tämä näkemys hyväksytään, voidaan hiljaisen tietämyksen empiiristä tutkimusta pitää äärimmäisen vaikeana, ellei jopa mahdottomana. Ambrosini ja Bowman [2001] pitävät empiiristä tutkimusta kuitenkin mahdollisena ja kuvailevat siihen sopivan tutkimusprosessin, jonka pääpiirteet käsittelen tässä luvussa.

Ambrosinin ja Bowmanin [2001] mukaan hiljaisen tietämyksen empiirinen tutkimusprosessi koostuu kahdenlaisista metodeista, jotka täydentävät toisiaan. Nämä metodit ovat toiminnan havainnointi ja kausaalikarttojen muodostaminen. Toimintaa voi havainnoida toimintaan itse osallistumalla tai ulkopuolelta tarkkailemalla. Tarkkailijalla voi olla mahdollista saada hiljaista tietämystä toiminnasta siihen itse osallistumalla. Toiminnan ulkopuolisen havainnoinnin tarkoituksena on pyrkiä ymmärtämään toimintaa, jotta pystytään havaitsemaan ja erittelemään erilaisia hiljaisia tekijöitä ja suhteita, mistä toiminta koostuu. Ambrosinin ja Bowmanin mukaan tämä on tärkeä osa empiiristä tutkimusta, mutta siinä on tiettyjä rajoitteita. Ensinnäkin, havainnointi on tutkimusmetodina hyvin hidasta. Toiminnan ymmärtäminen ja sen sisältämien osatekijöiden sekä suhteiden selvittäminen voi viedä suhteettoman paljon aikaa. Toiseksi, toimintaa tarkkailevat tutkijat ovat inhimillisesti rajoittuneita havainnoissaan. Varsinkin suurta ryhmää tarkkaillessa voi olla mahdotonta ymmärtää riittävästi toimintaan liittyviä lainalaisuuksia.

Tutkimusmetodeista toisena toimii kausaalikarttojen teko. Kausaalikartat ovat kognitiivisia karttoja, joiden avulla pyritään selvittämään yksilön toimintaan liittyviä entiteettejä ja niiden välisiä suhteita. Ambrosini ja Bowman [2001] käsittelivät kausaalikarttojen tekoa organisaation kilpailukyvyn taustalla piilevän hiljaisen tietämyksen tutkimuksessa, mutta mielestäni tutkimusprosessia voi soveltaa monenlaisen hiljaisen tietämyksen tutkimiseen.

Kausaalikartan tekoprosessi aloitetaan haastattelututkimuksella, jotta saataisiin esille yksilön tietämyksestä tutkimusalueeseen liittyviä käsitteitä ja avainsanoja. Haastattelut voidaan suorittaa kahdella eri tavalla: Michel Bougonin kehittämällä self–Q -haastattelumetodilla tai ohjatulla haastattelulla. Self–Q -metodissa haastateltava keksii täysin vapaasti itselleen kysymyksiä ennalta annetusta aiheesta. Kausaalikartan aloittamisessa käytetyt käsitteet ja avainsanat poimitaan haastateltavan keksimistä kysymyksistä ja vastauksista. Tämän metodin etuna nähdään, että itse tehdyt kysymykset eivät luo kynnystä vastaamista kohtaan ja ulkopuolisen kysyjän mahdollinen rajallinen tietämys aiheesta ei vaikuta haastattelun tulosten laatuun. Ohjattujen haastatteluiden tavoite ja rakenne on joiltakin osin ennalta määritelty – haastattelija antaa aiheen ja määrittää tavan, jolla haluaa vastauksia annettavan. Tarinat ja metaforat ovat

hyviä tapoja saada vastauksia, sillä niihin sisältyy usein kertojan hiljaista tietämystä. [Ambrosini and Bowman, 2001]

Haastattelujen avulla aihealueesta saadaan esiin käsitteitä ja avainsanoja, joiden avulla lähdetään rakentamaan kausaalikarttaa. Tutkittava asetetaan tarkastelemaan käsitteitä ja avainsanoja useasta eri näkökulmasta käyttäen hyväksi erilaisia kysymyksiä. Saaduista vastauksista eriytetään uusia käsitteitä, joita lähdetään jälleen avaamaan kysymysten avulla. Tutkittava aihe avataan näin yhä syvempiin ja yksityiskohtaisempiin kerroksiin. Esiin saadut käsitteet ja näiden väliset suhteet rakentavat kausaalikartan. Mitä syvemmälle kausaalikartan koostamisessa mennään, sen vaikeampaa voi tutkittavalle olla löytää kysymyksiin kerronnallisesti esitettäviä vastauksia. Tutkija voi tällöin pyytää vastauksia kielikuvien keinoin. Äärimmäisen tärkeää kausaalikartan rakentamisessa on, että tutkittavan vastaus kertoo, miten asiat hänen mielestään aktuaalisesti ovat. Vastausten ei pidä käsitellä sitä, miten asioiden tutkittavan mielestä pitäisi olla. [Ambrosini and Bowman, 2001]

Valmis kausaalikartta on ainakin osittainen kuvaus tutkittavan yksilön henkilökohtaisesta tietämyksestä – tutkittavan aihealueen entiteeteistä ja niiden välisistä suhteista hänen kokemanaan. Yksilön hiljaista tietämystä voi olla täten mahdollista saada esiin yhdistämällä kausaalikartan ja havainnoinnin tuloksia. Ryhmiä tutkittaessa tuloksia voidaan hakea yhdistelemällä yksilöiden tutkimustuloksia tai hakemalla niiden jonkinlaista keskiarvoa. Luotettavampi vaihtoehto on kuitenkin tutkia ryhmää yhdessä, jolloin ryhmän jäsenten väliset suhteet ja mahdollisen synergian vaikutus saadaan luotettavammin esiin. [Ambrosini and Bowman, 2001]

9. Omat loppumietteet

Aihetta tutkiessani oli mielenkiintoista havaita millaisena vedenjakaja Polanyin hiljaisen tietämyksen käsitteen tulkinta toimi. Jos Polanyin tulkitaan tarkoittavan, että hiljainen tietämys todella on näkymätöntä ja saavuttamatonta, niin mielestäni se ei kuitenkaan kumoa kokonaan Nonakan ja Takeuchin teorian lopputulemaa organisaatioiden tietämyksen luomisesta. Voidaan mielestäni perustellusti väittää, että organisaatioissa todella luodaan uutta eksplisiittistä tietämystä ja ihmisten oppiessa uusia taitoja, he sisäistävät ne osaksi mentaalisia mallejaan. Avoimeksi kysymykseksi tällöin jää, voiko hiljaista tietämystä jotenkin muuntaa tai siirtää eksplisiittiseksi tietämykseksi.

Käsite ryhmän hiljaisesta tietämyksestä oli myös mielenkiintoinen aihe pohdittavaksi. Lähinnä kysymys on mielestäni siitä, että voidaanko ryhmään kuvitella syntyvän jokin yhteinen, jaettu tietämys, joka selittäisi esimerkiksi ryhmässä syntyviä synergiaetuja. Vai voisiko kysymys olla ainoastaan siitä, että

ajan kanssa ryhmän jäsenet oppivat tuntemaan toisensa ja yhteisen toiminnan säännöt niin hyvin, että ryhmä toimii *kuin* omaisi jonkinlaisen jaetun tietämyksen. Tämäkin on mielestäni hyvin pitkälle tulkintakysymyksiä.

On mielestäni kuitenkin selvää, että tulkitaan teorioita miten tahansa, niin hiljainen tietämys on konkreettisesti olemassa oleva käsite. Ihmisen toimintaan ja tietämykseen liittyy varmasti paljon tekijöitä, joita emme pysty tietoisesti tunnistamaan. Oman näkemykseni siitä, voiko tätä hiljaista osaa tietämyksestämme jotenkin mahdollisesti aistia ja kommunikoida tietoisesti ulospäin, jätän odottamaan mahdollisia jatkoselvityksiäni.

Viiteluettelo

[Ambrosini and Bowman, 2001] Veronique Ambrosini and Cliff Bowman, Tacit knowledge: some suggestions for operationalization. *Journal of Management Studies* **38**, 6 (Sep. 2001), 811–829.

[Argote and Ingram, 2000] Linda Argote and Paul Ingram, Knowledge transfer: a basis for competitive advantage in firms. *Organizational Behavior and Human Decision Processes* **82**, 1 (May 2000), 150–169.

[Boisot and Canals, 2004] Max Boisot and Agusti Canals, Data, information and knowledge: have we got it right? *J. Evol. Econ.* **14** (2004), 43–67.

[Cook and Brown, 1999] S.D.N. Cook and J.S. Brown, Bridging epistemologies: the generative dance between organizational knowledge and organizational knowing. *Organization Science* **10**, 4 (1999), 381–400.

[Erden et al., 2008] Zeynep Erden, Georg von Krogh and Ikujiro Nonaka, The quality of group tacit knowledge. *Journal of Strategic Information Systems* **17** (2008), 4–18.

[Kangassalo, 2004] Hannu Kangassalo, Tiedon ominaisuuksista. Luonnos, Tietojenkäsittelytieteiden laitos, Tampereen yliopisto, marraskuu, 2004.

[Kolehmainen, 2001] Jari Kolehmainen, Yritykset ja alueet tietointensiivisessä globaalitaloudessa – kilpailukyky kohtalonyhteytenä. Tampereen yliopisto, Alueellisen kehittämisen tutkimusyksikkö, Sente-julkaisuja **12/2001**.

[Nonaka and Konno, 1998] Ibujiro Nonaka and Noboru Konno, The concept of "ba": building a foundation for knowledge creation. *California management review* **40**, 3 (1998), 40–54.

[Nonaka and Takeuchi, 1995] Ikujiro Nonaka and Hirotaka Takeuchi, *The Knowledge - Creating Company*. Oxford University Press, 1995.

[Tsoukas, 2002] Hardiamos Tsoukas, Do we really understand tacit knowledge? In: Mark Easterby-Smith and Marjorie A. Lyles, *Handbook of Organizational learning and Knowledge Management*, Blackwell Publishing, 2003, 410–428.

[Wilson, 2002] T.D. Wilson, The nonsense of 'knowledge management'. *Information Research* **8,** 1 (2002).

[Zins, 2007] Chaim Zins, Conceptual approaches for defining data, information and knowledge. *J. Am. Soc. Inf. Sci. Technol.* **58**, 4 (Jan. 2007), 479–493.

iPad aamiaispöydässä – tablettien ja sähköisten lehtien symbioosi

Paavo Virta

Tiivistelmä.
Tablet-laitteet ovat ottaneet vankan jalansijan tietotekniikkamarkkinoilla vajaassa kahdessa vuodessa. Samaan aikaan jo aiempien Internetin tuomien rakennemuutosten kanssa kamppailevat mediatalot ovat uusien haasteiden edessä, kun tabletit ovat sähköisten lehtien myötä mullistamassa sanoma- ja aikakauslehtien lukutottumuksia. Tässä tutkielmassa tarkastellaan tablettien ja sähköisten lehtien alkanutta yhteiseloa. Tutkielmassa luodaan katsaus niin tablettien historiaan, kosketusnäyttöjen toimintaan kuin median rakennemuutoksen syihin ja seurauksiinkin – tablettien kohdalla usein korostettua helppokäyttöisyyttä unohtamatta. Taustoituksen jälkeen tutkielmassa tarkastellaan tablettien ja niillä julkaistujen lehtien tämän hetkistä tilannetta ja siihen liittyviä oleellisia piirteitä. Lisäksi tutkimus pyrkii raottamaan verhoa tablettien ja sähköisten lehtien tulevaisuuden näkymiin.

Avainsanat ja -sanonnat: Tablet, iPad, sähköinen lehti, sähköinen julkaiseminen, kosketusnäyttö, käytettävyys, median rakennemuutos, sanoma- ja aikakauslehdet, journalismin tulevaisuus.
CR-luokat: H.5.1, H.5.2

1. Johdanto

Manuel Castells [1996] lanseerasi viime vuosituhannen loppupuolella termin informaatioyhteiskunta, joka kuvaa aikamme teknologisoitumista ja siirtymistä teollisuusyhteiskunnan jälkeiseen aikaan. Informaatioyhteiskunta nojaa sananmukaisesti tietoon, jonka lähteitä ovat pääosin Internet, media ja tietotekniikka sekä verkostoitumisen myötä lisääntynyt ihmisten välinen kommunikaatio.

Digitaalisen murroksen vauhti on vain kiihtynyt vuosituhannen vaihteen jälkeen. Samalla on muuttunut myös informaatioyhteiskunnassa elävien kuluttajien mediakäyttäytyminen. Internet on vakiinnuttanut asemansa erottamattomana osana arkea niin työvälineenä, viihdekeskuksena kuin tiedon lähteenäkin ja on teknologisten innovaatioiden myötä mahdollistanut eri medioiden käytön vuorokaudenaikaan tai paikkaan katsomatta.

Viimeisin suuren yleisön suosioon noussut innovaatio on ollut vuoden 2010 tammikuussa esitelty Apple iPad. Tabletiksi ristitty kosketusnäytöllä toimiva tietokone aloitti uudenlaisten tablet-laitteiden aikakauden, jonka päärоо-

leja näyttelevät juuri Internet, media ja tietotekniikka. Lyhyessä ajassa iPad on mullistanut tietotekniikan markkinoita lähes samassa mittakaavassa kuin IBM:n ja Applen ensimmäiset tietokoneet 1980-luvun alkupuoliskolla. Henkilökohtaiset tietokoneet ovat muuttumassa yhä selvemmin viestintälaitteiksi, joissa korostuvat helppokäyttöisyys ja erilaiset sähköiset sisällöt.

Tietotekniikan kehittymisen tuomien muutosten vaikutus on nähtävissä kaikilla elämän osa-alueilla aina yhteydenpidosta työelämän muutoksiin. Myös moni ala on joutunut uudistumaan radikaalisti tekniikan muuttaessa ihmisten tottumuksia ja markkinakäyttäytymistä. Uudistumisen paineet ovat 1990-luvulta lähtien kohdistuneet myös media-alaan ja etenkin lehtiteollisuuteen. Sanoma- ja aikakauslehtitaloilla on jo vuosia ollut käynnissä alamäki. Levikit ovat pienentyneet, mainostulot vähentyneet ja tilausmäärät kutistuneet. Internet on synnyttänyt ilmaisten uutisten ja vapaan median kanavan, joka on koukuttanut etenkin nuoret sukupolvet ja vienyt mainostuloja verkkoon. Median murros onkin pakottanut monet lehdet tehostamaan toimintaansa ja panostamaan printtimedian ohella verkkosisältöön – siitäkin huolimatta, että verkkotoiminta on usein tappiollista. [Pietilä, 2007]

Lehtitaloissa pöly ei ole kuitenkaan ehtinyt edes laskeutua Internetin tuomien rakennemuutosten jäljiltä, kun teknologian kehitys on jo aiheuttamassa uusia muutoksia. Suuren hypen saattelemana vuoden 2010 tammikuussa julkistettu, ja huhtikuussa markkinoille tullut, Applen iPad-laite on käynnistänyt väistämättömän sähköisten lehtien esiinmarssin [Heikkilä, 2011]. Tietä tableteille ovat olleet raivaamassa Kindlen kaltaiset lukulaitteet, jotka oli suunniteltu lähinnä yhtä käyttötarkoitusta varten – lukemiseen [Harju *et al.*, 2011]. Tablet-laitteet ovat vieneet pelin seuraavalle tasolle. Niissä yhdistyvät perinteisen tietokoneen ja älypuhelimen hyvät puolet. Helppokäyttöisten, alle kilon painoisten, 7-10 tuuman näytöllä varustettujen ja kosketuksella toimivien tietokoneiden menestyksekäs esiinmarssi onkin mullistamassa tapojamme käyttää Internetiä ja sosiaalista mediaa. Lisäksi tabletit ja niihin liittyvät sovelluskaupat tarjoavat laajat markkinat muun muassa sähköisille kirjoille ja lehdille.

Jo ennen iPadin lanseerausta lehtialalla spekuloitiin ajatuksella, että kyseinen laite saattaa toimia lehtialan pelastajana. Lehtikustantajien innostus vaihtui kuitenkin nopeasti skeptisyydeksi, kun tajuttiin, että sovellusten kehittämiseen kuluu aikaa ja rahaa [Vehkoo, 2011]. Tablettien vallankumouksellinen suosio on kuitenkin pakottanut monet mediat pohtimaan erillisen sähköisen lehden taittoa. Suomessakin lähes jokainen merkittävä lehti on ottanut vuoden 2011 aikana kantaa siihen, aletaanko julkaista tabletilla vai ei [Harju *et al.*, 2011]. Parin vuoden aikana onkin nähty paljon erilaisia ratkaisuja sähköisten sanoma-

ja aikakauslehtien taittamisessa. Samalla on esitetty myös ennustuksia printtimedian kuolemasta [Mayer, 2009].

Kosketusnäytöllä toimivia pieniä tietokoneita on ehditty nimittää monella tavalla. Esillä olleet ainakin termit taulutietokone, sormitietokone ja kämmentietokone. Tässä tutkielmassa käytetään tablet-termiä, joka on peruuttamattomasti vakiintumassa käyttöön ympäri maailmaa.

Tämä tutkimus tarkastelee tablettien ja sähköisten lehtien alkanutta maailmanvalloitusta. Ensiksi lähdetään liikkeelle historiasta ja taustoista: toisessa luvussa luodaan katsaus tablettien ja kosketusnäyttöjen historiaan sekä toimintaan. Koska tablettien kohdalla korostetaan monesti helppokäyttöisyyttä, niin kolmannessa luvussa pohditaan käytettävyyden käsitettä yleisellä tasolla. Neljännessä luvussa tarkastellaan tablettien markkinatilannetta. Viidennessä luvussa luodaan katsaus media-alalla käynnissä olevan rakennemuutoksen syihin ja seurauksiin. Kuudes luku tarkastelee lähemmin tablettien ja sähköisten lehtien symbioosia: mitä niiden välillä on tapahtunut, miten ne toimivat, mitä tekijöitä niihin liittyy, miten ne on otettu vastaan ja miltä niiden tulevaisuus näyttää. Seitsemäs luku sisältää yhteenvedon tutkielman sisällöstä ja loppupäätelmiä.

2. Kosketusnäyttölaitteiden kehitys

Tässä luvussa luodaan lyhyt katsaus tablet-laitteiden ja kosketusnäyttöjen historiaan sekä tarkastellaan markkinoiden yleisimpien kosketusnäyttöjen toimintaa.

2.1 Tablettien historia

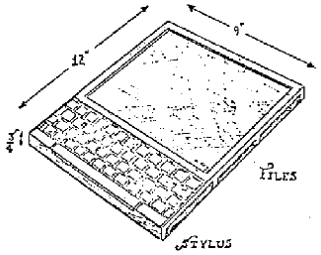
”On hulluutta tehdä sama asia uudelleen ja uudelleen, mutta odottaa eri tuloksia”, kirjoitti amerikkalainen kirjailija Lisa Mae Brown vuonna 1983 teoksessaan Sudden Death. Hänen ajatuksensa olikin vuosikymmeniä relevantti tablettietokoneiden markkinoilla. Valmistajat yrittivät vuosikymmeniä ajaa markkinoille tuotetta, jonka kuluttajat kerta toisensa jälkeen tyrmäsivät. Epäonnistuneiden tuotteiden lista ehti kasvaa pitkäksi, ennen kuin Apple osoitti vuoden 2010 tammikuussa, ettei yrittäminen ollut turhaa. Sittemmin iPadin myötä alkanut tablettien vallankumous on käynnistänyt uuden kehityssuunnan, josta tuskin on enää paluuta. Applen innovaatiolla ehti kuitenkin olla useita esikuvia vuosikymmenten varrella. Seuraavassa esitellään tabletteja, joiden läpimurto ei koskaan onnistunut, mutta joiden ominaisuudet eivät loppujen lopuksi olleet kovin kaukana tämän päivän käytössä olevista laitteista.

2.1.1 Dynabook

Jo vuonna 1968 Alan Key suunnitteli Dynabook-nimisen laitteen, jonka oli tarkoitus olla pieni ja kevyt, ensisijaisesti kaiken ikäisille lapsille suunnattu laite (ks. kuva1). Maailman ensimmäisen tabletin pääfunktio oli toimia monipuolisena työkaluna oppimisessa. Kay [1972] kutsuikin laitetta "kaiken ikäisten lasten henkilökohtaiseksi tietokoneeksi".

Dynabook oli levymäinen, noin tumman vahvuinen ja 12x9 tuuman näytöllä varustettu laite [Harju *et al.*, 2011]. Vaikka Kayn vallankumouksellinen visio ei koskaan nähnyt päivänvaloa konkreettisessa muodossa, niin hänen ajatuksensa ja innovaationsa olivat hyvin samankaltaisia kuin mitä iPad ja muut tabletit ovat pyrkineet hyödyntämään. Kayn yksityiskohtainen visio laitteen käytöstä on uskomattoman lähellä sitä, miten ihmiset käyttävät tabletteja tänä päivänä. Hän suunnitteli laitteen ulkoasua niin, että näyttöpaneeli kattaisi lähes koko sen etupuolen, jolloin virtuaalinen näppäimistö voisi asettua mihin kohtaan tahansa. Lisäksi hän hahmotteli laitteelle kykyä toimia suuren kaistanleveyden verkossa ja kykyä toistaa ääntä useita tunteja. Hän myös hahmotteli systeemiä, jossa käyttäjä voisi ladata kirjoja laitteeseen. Laitteen hinnaksi Kay suunnitteli 500 dollaria [Kay, 1972]

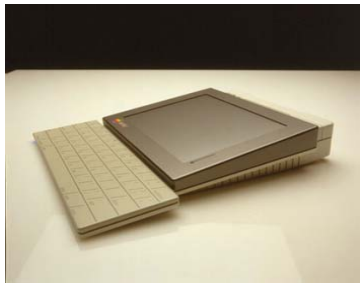
Xerox Palo Alton tutkimuskeskuksessa työskennelleen Alan Kayn visio ei koskaan päätynyt tuotantoon asti. Hänen nelisenkymmentä vuotta sitten laaditut suunnitelmansa kuvastavat kuitenkin melkoisesti niitä ominaisuuksia, joilla iPad löi itsensä läpi markkinoilla.



Kuva 1: Dynabookin hahmotelma vuodelta 1968.

2.1.2 Apple Bashful

Apple iPadin isoisä, Apple Bashful, suunniteltiin 27 vuotta ennen tablettien läpimurtoa. Laitteen prototyyppi valmistui vuonna 1983, mutta markkinoille se ei koskaan päätynyt. Varhaiseen tietokoneeseen suunniteltiin myös lisäosia, kuten näppäimistö, stylus eli osoitinkynä, levykeasema, puhelin ja kantolaukku (ks. kuva 2). [Steele, 2011]



Kuva 2: Apple Bashfulin prototyyppi kehitettiin vuonna 1983.

2.1.3 GRiDPad

Kosketusnäytöllä ja stylus-kynällä pääosin käytettävä Gridpad esiteltiin vuonna 1988. GriD-yhtymän suunnittelema kannettava laite sisälsi muun muassa tekstintunnistuksen, MS-DOS-käyttöjärjestelmän ja 10-tuumaisen mustavalkonäytön, jota ohjattiin styluksella. Se saavutti rajallisen suosion muutamissa instituutioissa Yhdysvalloissa, mutta noin 2370 dollaria maksanut laite oli useimmille liian kallis, eikä saavuttanut kaupallista menestystä (ks. kuva 3). [Steele, 2011]

Kuva 3: Gridpad julkaistiin vuonna 1988.

2.1.4 Freestyle

Wang Laboratories lanseerasi vuonna 1989 toimistokäyttöön tarkoitetun laitekokonaisuuden. Freestyle-nimeä kantanut paketti sisälsi tabletin, styluksen, ohjelmiston, puhelimen, faksin ja skannerin. Styluksen käyttöön pohjautunut kokonaisuus pyrki helpottamaan toimistotyöskentelyä. Sen suosio jäi kuitenkin vähäiseksi (ks. kuva 4). [Steele, 2011].

Kuva 4: Freestyle suunniteltiin ensisijaisesti toimistokäyttöön vuonna 1989.

2.1.5 AT&T Personal Communicator

Puhelin, modeemi, faksi, mikrofoni, kalenteri ja tekstinkäsittely olivat kaikki sisällytetty AT&T:n vuonna 1993 julkaisemaan laitteeseen (ks. kuva 5). Mielenkiintoisen oloinen AT&T Personal Communicator ei kuitenkaan koskaan menestynyt markkinoilla. Heikon menestyksen myötä laitteessa käytetty käyttöjärjestelmä ja mikroprosessorit tulivat tiensä päähän. [Steele, 2011]

Kuva 5: AT&T Personal Communicator

2.1.6 Apple Newton

Applen toinen tuleminen tablettirintamalla tapahtui vuonna 1993. Apple Newton-laitteessa oli muun muassa kosketukseen reagoiva näyttö (ks. kuva 6). Newton ei ollut markkinoilla menestys, mutta siitä tuli kuitenkin pitkäikäisempi kuin monista edeltäjistään. Vaikka alkuperäistä laitetta seurasivat useat uudet ja parannellut versiot, niin Newton ei kuitenkaan menestynyt odotetusti ja koko tuoteperhe ajettiin lopulta alas vuonna 1998. [Steele, 2011]

Kuva 6: Apple Newton

2.1.7 Microsoft Tablet PC

Vuonna 2000 ohjelmistoyhtiö Microsoft aloitti tablet-laitteen tuotekehityksen, joka huipentui kaksi vuotta myöhemmin yhtiön julkaistessa ensimmäisen tablettinsa. Windows XP-käyttöjärjestelmään ja styluksen käyttöön pohjautunut laite pyrki tuomaan käyttäjän ulottuville tavallisen tietokoneen toiminnot kos-

ketusnäytöllä varustetussa, mukana kulkevassa laitteessa (ks. kuva 7). Samalla kyseinen laite oli ensimmäinen, jonka kohdalla käytettiin tablet-termiä. [Steele, 2011]

Microsoftin toimitusjohtaja Bill Gates ennakoi vuonna 2002, että tableteista tulee viiden vuoden kuluessa kaikkein suosituin tietokonetyyppi. Valtavan markkinoinnin saattelemana julkaistun tuotteen myötä näyttikin hetken siltä, että tablettien aikakausi olisi alkanut. Uudet laitteet osoittautuivat kuitenkin kömpelöiksi ja suorituskyvyltään vaatimattomiksi, eikä Microsoftin innovaatio onnistunut tavoittamaan suurta yleisöä. Kuluttajat käyttivät mieluummin kannettavia tietokoneita ja tablettien läpimurto antoi vielä odottaa itseään. [Evans, 2011]

Kuva 7: Microsoft Tablet PC lanseerasi vuonna 2002 tablet-termin.

2.1.8 Apple iPad

Apple onnistui vuonna 2010 siinä, missä kukaan muu ei ollut onnistunut aiemmin – se toi markkinoille menestyvän tablet-tietokoneen (ks. kuva 8). Kun iPad julkistettiin, oli tabletteja kohtaan kertynyt odotuksia jo vuosikymmenten ajan [Harju *et al.,* 2011]. Applen tablet-laite käynnisti uuden henkilökohtaisten tietokoneiden kehityssuunnan ja muut valmistajat alkoivat nopeasti tuoda omia tablettejaan markkinoille.

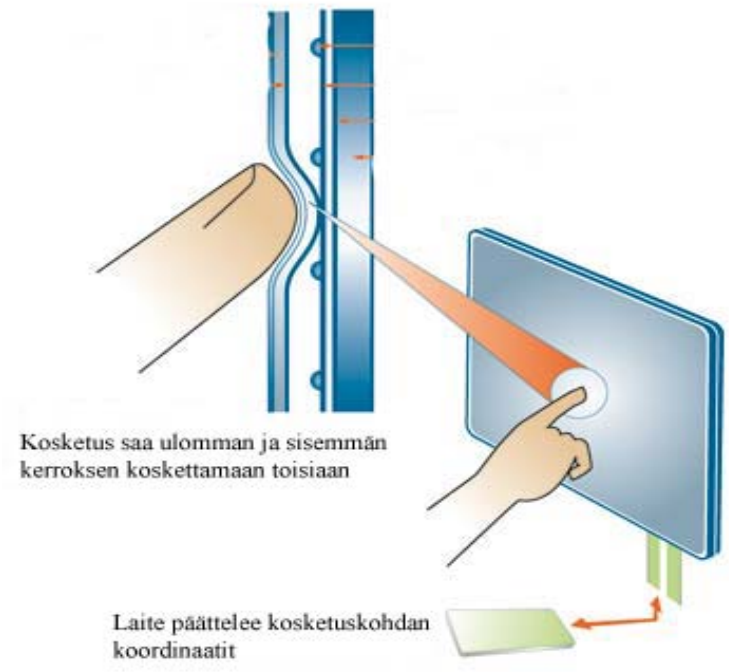
Kuva 8: Vuonna 2010 julkaistu iPad on ensimmäinen markkinoilla menestynyt tablet-laite.

2.2 Kosketusnäytöistä

Kannettavien mobiililaitteiden, älypuhelinten ja tablettien, oleellinen osa on kosketukseen nopeasti ja vaivattomasti reagoiva näyttö. Kosketusnäytöillä on kuitenkin takanaan jo vuosikymmenten historia ja kehitystyö. Seuraavaksi tarkastellaan lyhyesti kosketusnäyttöjen historiaa ja perehdytään hieman tämän päivän suosituimpien tekniikoiden toimintaan.

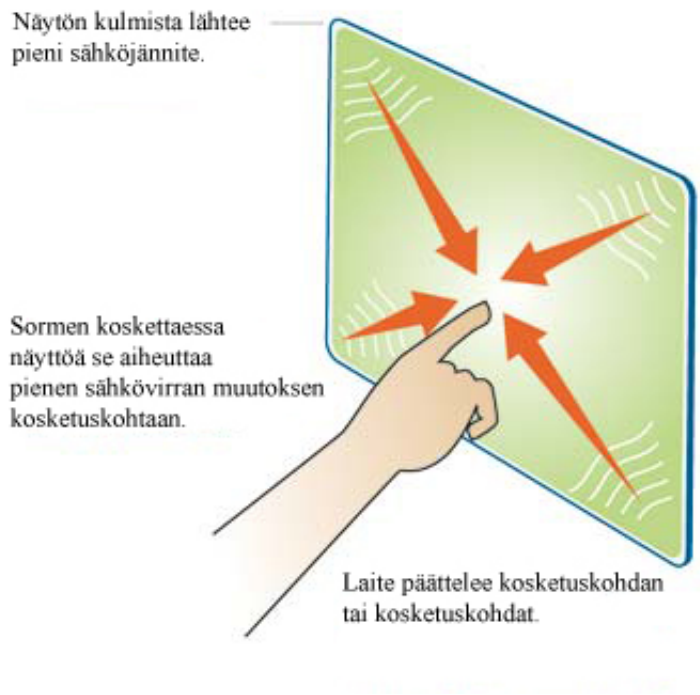
Ensimmäisten kosketusnäyttöjen kehitys alkoi yli 40 vuotta sitten. Kentuckyn yliopiston opettaja Sam Hurst kehitti vuonna 1971 ilmestyneen ensimmäisen kosketusnäytön. Kosketussensori kantoi nimeä Elograph Hurstin perustaman Elographics-yhtiön mukaan. Kyseinen näyttö ei vielä ollut läpinäkyvä, mutta vuonna 1974 Hurst ja Elographics-yhtiö kehittivät ensimmäisen läpinäkyvän kosketusnäytön. Kolme vuotta myöhemmin samat ihmiset olivat kehittämässä resistiivistä kosketusnäyttöteknologiaa, joka on edelleen suosituimpia teknologioita kosketusnäyttölaitteissa. [Bellis, 2008]

Merkittävimmät tänä päivänä käytössä olevat kosketusnäyttöteknologiat ovat resistiivinen ja kapasitiivinen. Resistiivinen kosketusnäyttö perustuu kahteen sähköä johtavaan läpinäkyvään ja joustavaan kalvoon, joiden väliin jää eristävä rako (ks. kuva 9). Kosketettaessa näyttöä kalvot painautuvat kosketuskohdasta yhteen, jolloin sähkövirta pääsee kulkemaan niiden välillä, ja laite tunnistaa kosketuskohdan. [Hamilo, 2010]



Kuva 9: Resistiivinen kosketusnäyttö

Kapasitiivinen kosketusnäyttö koostuu sähköeristeestä, kuten lasista, joka on päällystetty sähköä johtavalla aineella. Ihminen johtaa sähköä, joten näytön koskettaminen aiheuttaa muutoksen näytön sähkökenttään, mistä laite voi päätellä kosketuskohdan (ks. kuva 10). Valtaosa kalliimman luokan kosketusnäyttölaitteista on varustettu kapasitiivisella näytöllä. Syynä on se, että kapasitiivinen näyttö reagoi sormenpään kevyeenkin kosketuksen ja tekee sen resistiivistä näyttöä nopeammin. Siksi monet käyttäjät kokevat sen miellyttävämmäksi käyttää. Toisaalta kapasitiivinen näyttö ei reagoi kynnellä tai stylus-kynällä painamiseen toisin kuin resistiivinen näyttö. [Hamilo, 2010]



Kuva 10: Kapasitiivinen kosketusnäyttö.

Kosketusnäyttöjen yksi vahvuus on niiden mukautuminen tilanteiden mukaan. Käyttäjälle voidaan näyttää vain ne painikkeet, joita kussakin tilanteessa on mahdollista painaa [Hyyrysalo, 2009]. Tällöin tilanteen kannalta turhat vaihtoehdot eivät vie tilaa laitteessa. Innovatiiviset usean sormen yhtäaikaista kosketusta tukevat kosketustavat on jo todettu varsin käyttäjäystävällisiksi. Pyyhkäisy eli sormen vetäminen näytöllä, nipistys tai sormien levittäminen eli ulos- tai sisäänpäin zoomaus sekä pelkkä halutun kohteen koskettaminen näytöllä ovat muutamia jo vakiintuneita kosketusnäytön käyttötapoja. Niiden avulla on luonnollista kommunikoida sovellusten kanssa tai vaikkapa navigoida sähköisen lehden sivuilla.

Koskettaminen on luonnollinen ja inhimillinen kommunikointitapa. Applen edesmennyt toimitusjohtaja Steve Jobs totesi ensimmäisen iPhonen julkai-

sutilaisuudessa, että kosketusnäyttölaitteissa kannattaa käyttää välinettä, jonka käyttöä ihminen on opetellut koko elämänsä – sormea. Jopa osoittamiseen tarkoitettu stylus-kynä on liikaa [Jobs, 2007]. Kosketusnäyttöjen uusi tuleminen alkoi juuri iPhonen myötä älypuhelimista ja on räjähdysmäisesti levinnyt myös tablet-laitteisiin.

Tutkimusyhtiö Gartner arvioi hiljattain, että viiden vuoden kuluttua yli puolet alle 15-vuotiaille hankittavista tietokoneista toimii kosketusnäytöllä. Gartnerin mukaan laitteet menevät ensisijaisesti viihde- ja opetuskäyttöön. Työelämässä ne sen sijaan tulevat yleistymään hitaammin. Tulevaisuus näyttää, opitaanko kosketusohjausta hyödyntämään entistä innovatiivisemmin ja tehokkaammin. Jos näin tapahtuu, saattaa olla, että kymmenen vuoden kuluttua näppäimistöt ja hiiret alkavat olla taakse jäänyttä elämää. [Järvinen, 2010]

3. Käytettävyydestä

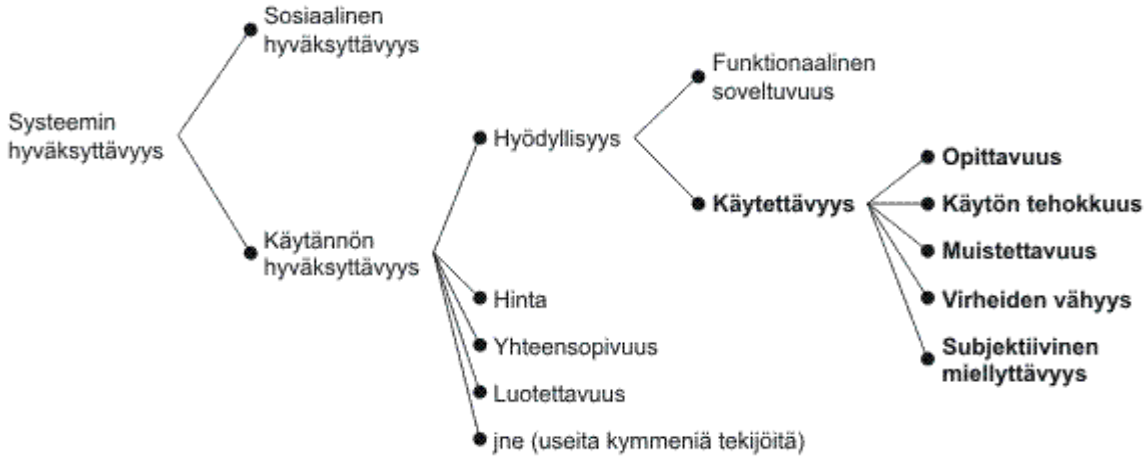
Tableteista puhuttaessa mainitaan monesti niiden olevan helppokäyttöisiä. Laitteiden ja sovellusten helppokäyttöisyys onkin yksi tärkeimmistä tekijöistä kuluttajalle. Visuaalisesti kauniiden ja helppokäyttöisten käyttöliittymien kehitykseen panostaminen on yksi viimeaikaisista trendeistä ohjelmistoalalla ja se kuvastanee luultavasti myös tulevaisuuden suuntaa – helppokäyttöisyys ja visuaalisuus ovat nousemassa yhä keskeisempään asemaan. Vaikka tämä tutkimus ei suoranaisesti tarkastele tablettien ja sähköisten lehtien käytettävyyttä, niin aihetta sivutaan useaan otteeseen. Sen vuoksi seuraavaksi tarkastellaan käytettävyyden käsitettä.

Käytettävyydellä tarkoitetaan yleensä helppokäyttöisyyttä ja helppoa opittavuutta [Preece, 1995]. Nielsenin [1993] mukaan käytettävyys tarkoittaa sitä, kuinka hyvin jonkin järjestelmän toimintoja voidaan käyttää haluttuun tarkoitukseen. Järjestelmän käytettävyyden taustalla ovat järjestelmän toiminnot, jotka määräävät sen hyödyn. Tällöin käytettävyys kertoo, kuinka onnistunutta näiden toimintojen käyttö on.

Käytettävyyteen kuuluu järjestelmän hyväksyttävyys, joka jakautuu kahteen osaan: sosiaaliseen ja käytännön hyväksyttävyyteen. Yhteensopivuus, luotettavuus, hyödyllisyys ja käyttökelpoisuus sekä kustannukset ovat käytännön hyväksyttävyyden osa-alueita. [Nielsen, 1993]

Nielsen [1993] on kehittänyt ominaisuuksien mallin käytännön hyväksyttävyydestä ja hyödyllisyydestä (ks. kuva 11). Mallissa käytettävyys jaetaan opittavuuteen, käytön tehokkuuteen, muistettavuuteen, virheiden vähäisyyteen ja subjektiiviseen miellyttävyyteen. Opittavuus tarkoittaa, että järjestelmän käyttö on helppo oppia ja käyttäjä pääsee mahdollisimman nopeasti haluamiinsa päämääriin. Käytön tehokkuus tarkoittaa, että käyttäjän opittua järjestelmän

käytön, hän pystyy hyödyntämään sitä tehokkaasti. Muistettavuudella tarkoitetaan sitä, että käyttäjä pystyy tauonkin jälkeen hyödyntämään järjestelmää ilman uudelleen opettelua – järjestelmän käyttö on helposti muistettavissa. Virheiden vähäisyys liittyy siihen, ettei valmiissa järjestelmässä esiinny virheitä. Virhetilanteen sattuessa järjestelmän tulee tukea käyttäjää. Subjektiivinen miellyttävyys tarkoittaa, että järjestelmä on miellyttävä käyttää, jolloin käyttäjät ovat siihen tyytyväisiä.



Kuva 11: Nielsenin [1993] malli järjestelmän hyväksyttävyydestä ja käytettävyydestä sen osana.

Käytettävyyden käsitettä voidaan tarkastella myös sitä määrittelevän ISO 9241-11 -standardin kautta. Kyseisessä standardissa määritellään käytettävyys ja esitetään, mitä tietoja tarvitaan näyttöpäätteiden ja tietojärjestelmien käytettävyyden määrittelemiseen ja arviointiin, kun mitataan käyttäjän suoriutumista ja tyytyväisyyttä. ISO 9241-11 määritelmän mukaan käytettävyydellä tarkoitetaan sitä vaikuttavuutta, tehokkuutta ja tyytyväisyyttä, jolla määritellyt käyttäjät saavuttavat halutut tavoitteet käyttötilanteessa. [ISO 9241-11, 1998; Sampola, 2008]

4. Tablettien markkinatilanne

Ensimmäisen iPadin lanseeraus vuoden 2010 tammikuussa osoitti, että tableteille on kysyntää. Pienikokoinen kosketusnäytöllä varustettu tietokone myytiin loppuun jo ennen sen saapumista markkinoille. Kun iPad maaliskuussa löysi tiensä kauppoihin, myytiin sitä ensimmäisten kahdeksankymmenen päivän aikana yli kolme miljoonaa kappaletta. [Apple Press Info, 2010]

Oxfordin yliopiston Reuters-instituutissa tabletteja ja sähköisiä lehtiä tutkiva Jussi Ahlroth [2011] on kiteyttänyt iPadin menestyksen kahteen seikkaan. Ensinnäkin iPad onnistui irtautumaan aiemmista rajoituksista, jotka kulminoi-

tuva käyttöliittymien paperi ja kynä -metaforaan. Aiempien, lähinnä Windows-pohjaisten tablettien, yksi suurimmista ongelmista oli ollut, että niiden käyttöjärjestelmä oli suunniteltu käytettäväksi tarkan osoitinlaitteen – hiiren tai styluksen – ja näppäimistön avulla. Näiden sijaan iPad tarjosi helppokäyttöisen ja toimivan kosketuskäyttöliittymän, jota operoitiin sormella, ja jota tuki sille optimoitu käyttöjärjestelmä.

Lisäksi Ahlroth [2011] näkee menestyksen takana olevana tekijänä uuden sovelluksiin pohjautuvan ekosysteemin. Applen sovelluskaupassa on tarjolla suunnaton määrä kaikkien elämänalueiden sovelluksia, jotka ovat helppokäyttöisiä ja edullisia. Sovellusten – tai ”appien” – ostaminen ja lataaminen on pyritty tekemään helpoksi. Sovellusten tuoma lisäarvo tableteille perustuu siihen, että ne tuovat valtavasti lisää käyttötarkoituksia laitteelle. Lisäksi ne ovat täysin käyttäjän itsensä valittavissa – käytännössä jokainen käytössä oleva tabletti on käyttäjänsä personoima ja ainutlaatuinen laite.

Myös muut tietotekniikkavalmistajat ovat huomanneet tablettien suuren kysynnän. Apple iPadin merkittävimmäksi haastajaksi on noussut eteläkorealaisen Samsungin Galaxy Tab-laite (ks. kuva 13). Sen ensimmäinen versio ilmestyi markkinoille marraskuussa 2010.

Kuva 13: Samsung Galaxy Tab-laitteen kaksi erikokoista versiota.

Toinen potentiaalinen haastaja on Asus Transformer -tabletti, joka on tabletin ja kannettavan tietokoneen yhdistelmä (ks. kuva 14). Halpojen tablettien ykköseksi on nousemassa Amazon Kindle Fire, joka on hinnoiteltu erittäin kilpailukykyisesti (ks. kuva 15). Amazon on kuitenkin saatavilla ainoastaan Yhdysvalloissa. Lukuisat tietotekniikkavalmistajat ovat tuoneet markkinoille omat tablet-laitteensa ja myös suomalaisen matkapuhelinvalmistajan, Nokian, on arveltu lähtevän mukaan tablettikilpailuun vuonna 2012 [Garside, 2011].

Kuva 14: Asus Transformer

Tablettien markkinat ovat mielenkiintoisessa tilanteessa, sillä kasvumahdollisuudet ovat suuret. Lyhyellä aikavälillä markkinat voivat kuitenkin olla hyvin vaikeat uudelle tulokkaalle, sillä Apple ja Samsung dominoivat tällä hetkellä kalliimpia tuotteita ja Amazonin kilpailukykyinen hinnoittelu on syönyt muiden myyntiä Yhdysvalloissa. Markkinoilla onkin jo uutisoitu, että Hewlett-Packard, Acer, Asustek ja Dell jättäisivät tablettien valmistamisen vuonna 2012. [Lee and Tsai, 2011]

Kuva 15: Amazon Kindle Fire

Kilpailua käydään niin näytön koon, laitteen laadukkuuden, suorituskyvyn, käyttöjärjestelmän, sovellusten kuin hinnankin ehdoilla. Kun tietokone koostuu lähes pelkästään näytöstä, on sen laadukkuus elinehto. Apple on väritoistoltaan ja katselukulmaltaan loistava, samoin Samsungin Galaxy Tab. Halpojen tablettien näytöt ovat usein haaleita ja katselukulmaltaan heikompia. Tabletit eivät näy suorassa auringonvalossa hyvin, mutta hieman suojaan käännettynä parhaiden näyttöjen käyttö onnistuu ulkona kirkkaanakin päivänä. [Bershewsky *et al.*, 2011]

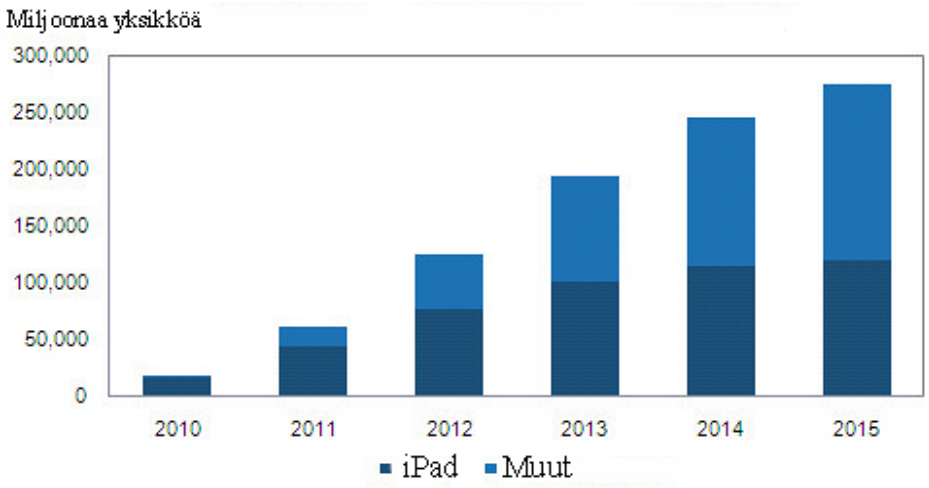
Markkinoiden suosituimmat käyttöjärjestelmät ovat tällä hetkellä Applen iOS ja Googlen kehittämä Android [Kauppalehti, 2011]. Vaikka Apple on edelleen ylivoimainen markkinajohtaja niin myyntitilastoissa kuin sovellusten kehitysalustanakin, niin Android on nopeasti noussut potentiaaliseksi kilpailijaksi. Tulevaisuudessa myös Microsoft on Windows-käyttöjärjestelmineen pyrkimässä mukaan peliin.

Älypuhelinten ja tablettien markkinoihin vaikuttaa yhä enenevissä määrin sovelluskauppa ja ohjelmistotarjonta, sillä markkinoiden teknisesti hienoinkin laite tuntuu turhalta ilman sisältöä – ja sisältöä tableteille riittää. Apple Store ja Android Store ovat tällä hetkellä laajimmat sovelluskaupat tablet-laitteille. Molemmista löytyy jo kuusinumeroinen luku sovelluksia. Applen musiikki- ja sovellustarjonta on ainakin toistaiseksi maailman kattavin. Kiinnostava uutinen mediataloille lienee se, että sovellusten uutiskategoria on jo nyt sekä kattava että suosittu. Tulevaisuus näyttää kuka lopulta selviää voittajana tablettisodassa. Käyttökokemus eli toisin sanottuna kuluttajat ratkaisevat miten lopulta käy. [Heikkilä, 2011; Ruulio, 2011]

Eräs tablettien suosiota selittävä tekijä on se, että laitteet ovat monilla käyttäjillä vieneet perinteisen kannettavan tietokoneen paikan. Nielsen Companyn [2011] tekemän tutkimuksen mukaan tärkeimpiä syitä tähän ovat tabletin helppo kuljettaminen mukana, helppokäyttöisyys ja laitteen viiveetön käynnistyminen.

Tablettien myyntivauhdissa ei näy hidastumisen merkkejä. Vuoden 2011 kolmannen neljänneksen aikana laitteita myytiin 16,7 miljoonaa kappaletta, mikä tarkoittaa 280 prosentin kasvua edellisen vuoden vastaavaan ajankohtaan verrattuna. Markkinoiden valtias on edelleen Applen iPad ja iOS- käyttöjärjestelmä, joiden prosentuaalinen osuus on kuitenkin laskenut. Applen laitteita toimitettiin 11,1 miljoonaa kappaletta, mikä on 66,6 prosenttia markkinoiden kaikista laitteista. Markkinaosuus on laskenut vuodessa 28,9 prosenttiyksikköä. Sitä ovat kahmineet Samsungin johdolla Android-käyttöjärjestelmään pohjautuvat laitteet, joiden markkinaosuus on noussut runsaasta kahdesta prosentista 27 prosenttiin. [Kauppalehti, 2011]

Tablettitietokoneiden myynnille odotetaan suurehkoa kasvua myös lähivuosina. Tutkimusyhtiö Euromonitor arvioi, että Yhdysvalloissa myydään 44 miljoonaa tablettia vuonna 2014. Myös markkinatutkimusyhtiö IHS [2011] ennustaa tablettien maailmalaajuisten toimitusmäärien kasvavan reilusti lähivuosien aikana. Yhtiö ennustaa iPadin pysyvän markkinajohtajana, vaikka muut valmistajat alkavatkin hiljalleen saada jalansijaa markkinoilla yhä enemmän (ks. kuva 16).



Kuva 16: Ennuste tablettien toimitusmääristä maailmanlaajuisesti [IHS, 2011]

5. Muuttuvat mediamarkkinat

Tiedotusvälineiden keskinäinen työnjako on muuttunut viimeisten vuosikymmenien aikana nopeasti Internetin ja muun sähköisen viestinnän kehittymisen myötä. Sähköistyminen asettaa ennennäkemättömiä haasteita koko mediakentälle. Internetin – ja etenkin laajakaistan – yleistymisen jälkeen ovat median sisällön muutokset olleet rajuja. Markkinat muuttuvat jatkuvasti tuotteiden ja kuluttajien siirtyessä uusin jakelukanaviin. Rakennemuutoksen kourissa kamppailevat ennen kaikkea painetut mediat, sanoma- ja aikakauslehdet, joiden uusin haaste on mukautua tablettien ja muiden sähköisten lukulaitteiden tuomiin muutoksiin. [Pietilä, 2007].

Tässä luvussa tarkastellaan median rakennemuutoksen syitä ja seurauksia.

5.1 Internet ja median rakennemuutos

Pietilän [2007] mukaan mediaan liittyvän kehityksen valtavirta kulkee Internetin mukana ja kaikki kulutuksen muutokset liittyvät jollakin tavalla siihen. Kehityskulku on ollut nähtävillä kaikkein selvimmin laajakaistan valtaamilla alueilla ja etenkin läntisissä teollisuusmaissa muuttuvat kulutustottumukset ovat nähtävissä hyvin konkreettisesti.

Uuden tekniikan tuoma murros on näkynyt etenkin paperin tietoarvon laskuna. Kaikesta uudesta tiedosta paperiin painettuun muotoon päätyy enää 0,01 prosenttia – suurin osa päätyy vain sähköiseen muotoon. Tämä kehityskulku on vakava paikka etenkin sanomalehdille. Vuodesta 1990 vuoteen 2007 sanomalehtipaperin kulutus laski kolmanneksen.

Laajakaistan yleistyessä salamavauhtia voidaankin miettiä, mihin tulevaisuus lopulta johtaa. Paperisten lehtien levikit ovat kääntyneet vakaaseen laskuun. Esimerkiksi Yhdysvalloissa paperilehtien tilaaja- ja lukijamäärät laskivat vuonna 2006 nopeammin kuin kertaakaan aiemmin 15 vuoden aikana. Joka viides laajakaistan hankkinut kotitalous perui USA:ssa sanomalehden tilauksen vuosina 2001-2006. Verkkofoorumien ja yksityisten uutissivustojen tarjoamat ilmaiset uutiset ovat omalta osaltaan vaikuttamassa siihen, että maksullisten lehtien suosio vähenee entisestään. Kansallisen mediatutkimuksen tekemän yksityiskohtaisen tarkastelun mukaan suuret suomalaiset sanomalehdet seuraavat yllättävän tarkasti Yhdysvaltojen kehityskulkua. Levikkien laskun on havaittu olevan melko suoraan yhteydessä laajakaistan leviämiseen. [Pietilä, 2007]

Verkko on lehdille kaksijakoinen paikka. Toisaalta se moninkertaistaa lehden tavoittamien ihmisten määrän, mutta toisaalta se syö lehden levikkiä ja tuloja. Esimerkiksi ruotsalaisen Aftonbladet-iltapäivälehden verkkotoimintoja on kehitetty voimakkaasti 2000-luvulla. Verkon petollisuus paljastui lopulta vuonna 2007, kun painetun lehden levikki ja taloudellinen tulos romahtavat, vaikka verkkokävijöiden määrä saavutti ennätyksen. Tilanne on samantyyppinen suomalaisilla lehdillä. Yksikään suurimmista sanomalehdistä ei ole kyennyt lisäämään lukijamääriään 1990-luvun lopun jälkeen. Ongelman ydin on mainostulojen dramaattinen lasku. Vaikka verkkomainonta on kasvanut valtavasti, niin sen hintataso on alhainen verrattuna printtiversioon. Verkkomainonnan ongelmana on, että Internetin mainosten seuraaminen riippuu pitkälti kuluttajien omasta aktiivisuudesta. [Pietilä, 2007]

Uudistumisen paineiden lisäksi lehtien kustantajia piinaavat kannibaalit ja loiset. Kannibaaleilla tarkoitetaan ilmaista nettisisältöä, joka syö – tai kannibalisoi – paperituotteen. Loiset puolestaan ovat juuri Google Newsin kaltaisia aggregaatteja, jotka levittävät toisten luomaa sisältöä verkossa. Ekonomistien mukaan on olemassa kahdenlaista kannibalisointia. Kun yritys tuo markkinoille tuotteen, joka vähentää sen vanhan tuotteen myyntiä, puhutaan suunnitelmallisesta kannibalisaatiosta. Suunnittelematonta kannibalisaatiota puolestaan tapahtuu, kun myyntitulot putoavat samasta syystä, mutta sitä ei ole osattu odottaa. Juuri edellä mainitut seikat ovat osaltaan vaikuttamassa siihen, että mediayhtiöt hidastelevat digitaalisessa tuotekehityksessä. [Vehkoo, 2011]

Vaikka verkko on elintärkeä toimintaympäristö monelle medialle, niin sisällöntuotantoineen ja tuotekehityksineen se on usein tappiollista toimintaa mediayhtiöille. [Pietilä, 2007]. Lisäksi tabletit ovat tulleet keskelle tilannetta, jossa monet toimitukset ovat juuri saaneet järjesteltyä suurin piirtein toimivat työnkulun paperilehdille ja verkkojulkaisemiselle. Nyt tarvittaisiin jälleen uutta organisointia. Vehkoon [2011] mukaan pelko onkin johtanut hidasteluun me-

diataloissa. Myös Ruulio [2011a] kirjoittaa monien sanomalehtien reagoivan tablettien tuomaan uuteen tilanteeseen ”kuin pupu, joka pisti pelosta päänsä pensaaseen”. Tilanne ei sinänsä ole uusi. Kolmekymmentä vuotta sanomalehtien uudistusten parissa työskennellyt Mario Garcia toteaa, että samanlaista hidastelua on nähty aiemminkin. Kun värit tulivat sanomalehtiin, monet vastustivat niitä, koska heidän mukaansa ne saivat lehdet näyttämään halvoilta. [Vinter, 2011] Myös analyytikko Ken Doctor on sitä mieltä, että mediatalot ovat olleet myöhässä viime vuosikymmenien jokaisessa digitaalisessa vallankumouksessa – esimerkiksi verkkojulkaisujen hakutoimintojen, videon, sosiaalisen median ja mobiiliympäristön hyödyntämisessä. [Harju *et al.,* 2011]

5.2 Kohti digitaalista tulevaisuutta

Vehkoon [2011] mukaan sanomalehdillä menee kuitenkin Suomessa vielä toistaiseksi kohtalaisen hyvin. Monet tahot ovat jopa miettineet, kannattaako lehteä edes viedä tabletille [Itkonen, 2011]. Totuus ja sanoma- ja aikakauslehtien tilanteessa kuitenkin on, että lehtien levikit ovat jo vuosia olleet vakaassa, pysäyttämättömässä laskussa. Uskollinen tilaajakanta vanhenee koko ajan, samalla kun nuoremmat polvet lukevat suurimman osan uutisistaan verkosta. Pienenevät rahavirrat ovatkin jo aiheuttanut säästötoimenpiteitä, ja vaikka suomalaisen lehdistön tilanne on parempi verrattuna moniin muihin länsimaihin, niin suomalaiset mediatalot ovat jo alkaneet supistaa toimittajien määrää sekä toimitusten taloudellisia resursseja. Samalla on osittain unohdettu, että journalismi kulkee yhä vahvemmin kohti digitaalista tulevaisuutta.

Levikkien pieneneminen on alkanut jo ennen tablettien ja niille tehtyjen sähköisten lehtien läpimurtoa. Samoin ennusteita sanomalehtien heikosta tulevaisuudesta on esitetty jo pidemmän aikaa. Synkimmät ennustajat ovat arvioineet, että sanomalehdet lakkaavat ilmestymästä paperisessa muodossa lähivuosikymmeninä. Journalismin professori Philip Mayer [2009] arvioi, että yhdysvaltalaiset sanomalehdet lopettavat ilmestymisensä vuoden 2043 ensimmäisellä neljänneksellä. Yhtenä merkittävänä syynä tähän ovat hänen mukaansa ikäluokkien erilaiset tottumukset. Yhä useammat nuoret aloittavat päivänsä selaamalla internetin verkkopalveluja sanomalehtien sijaan. Suomalaisista 15-24-vuotiaista nuorista 53 prosenttia nimesi Internetin mediaksi, josta ei voisi luopua [eMedia, 2007]. Tutkimuksissa onkin selvästi havaittavissa nuorten sukupolvien arvostus Internetiä kohtaan, mikä samalla antaa kuvan tulevaisuuden suunnasta. Vuonna 2011 tehdyssä tutkimuksessa Internet nousi ensimmäistä kertaa koko väestön tärkeimmäksi mediaksi – 15-79-vuotiaista vastaajista 38% ilmoitti Internetin olevan media, josta he eivät voisi luopua. [eMedia, 2011]

Jos amerikkalaisten asenteista voidaan tehdä minkäänlaisia johtopäätöksiä, niin lehtien tabletille vientiä on turha edes pohtia. Googlen omistaman mobiilimarkkinoinnin verkoston, AdMobin, tutkimuksen mukaan peräti 61 prosenttia amerikkalaisvastaajista kertoo pitävänsä tablettia sanomalehden korvikkeena [Itkonen, 2011]. Jos yli puolet ihmisistä haluaa lukea lehtensä tabletilta, on se perinteiselle printtimedialle haastava tilanne – lehtien on löydettävä roolinsa uudestaan. [Vehkoo, 2011]

Digitaalisen tulevaisuuden päärooleja näyttelevät mitä suurimmalla todennäköisyydellä juuri tabletit ja niihin tehdyt sähköiset lehdet. Michiel Buitelaar [2010] Sanoma Magazines -yhtymästä uskoo tablettien kiihdyttävän verkossa toimivan median muutosta. Hän toteaa, että iPad on ensimmäinen laite, josta löytyvät heidän aikakauslehtiensä verkkojulkaisussa kaivatut ominaisuudet: väri, kosketus, yhteydet ja vuorovaikutus – sekä lisäksi videoiden ja äänen toisto yhdessä tyylikkäässä ja helppokäyttöisessä paketissa. Sanoma Magazines kustantaa satoja lehtiä Euroopassa. Verkossa jo aiemmin julkaistuista lehdistä on päätelty, ettei pelkkä aikakauslehden näköisversion vieminen tabletille riitä. Sen vuoksi iPadille suunnitellaankin julkaisuja, jotka pyrkivät hyödyntämään tablettien ominaisuuksia: henkilökohtaisuutta, kannettavuutta, värejä, kehittynyttä kosketusnäyttöä, verkkoyhteyttä ja multimediaa sekä interaktiivisuutta. [Buitelaar, 2010]

6. Tablettien ja sähköisten lehtien symbioosi

Vielä muutama vuosikymmen sitten toimistoissa luotettiin lähinnä paperiseen materiaaliin. Sittemmin tietokoneet ovat mullistaneet toimistotyöskentelyn. Nyt edessämme näyttäisi olevan aika, jolloin sanoma- ja aikakauslehdille käy samoin kuin toimistotyölle – materiaali siirtyy paperilta verkkoon.

Symbioosi on biologiasta tuttu termi, jolla viitataan kahden eliölajin läheiseen yhteiseloon, josta molemmat hyötyvät. Tässä luvussa tarkastellaan tablettien ja sähköisten lehtien yhteiseloa, joka tarjoaa mahdollisuuksia molemmille osapuolille.

6.1 Sähköisten lehtien markkinatilanne

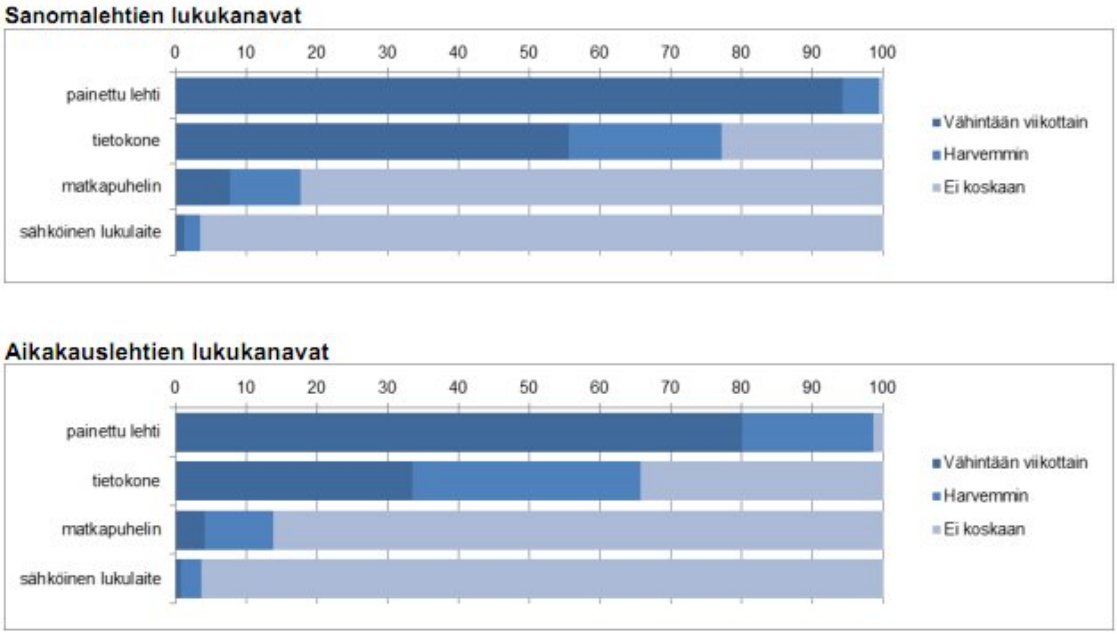
Tablet-julkaisujen aikakausi alkoi huhtikuussa 2010, kun iPad tuli markkinoille Yhdysvalloissa. Ensimmäisenä iPad-versionsa toivat markkinoille amerikkalaiset aikakauslehdet Wired, Popular Science ja Time. Kuluttajien kiinnostus oli heti ilmeistä. Esimerkiksi Wired-lehteä myytiin ensimmäisellä viikolla 73 000 ja ensimmäisen kuukauden aikana yli 100 000 kappaletta. Zinio-julkaisualustaa tuottava taho arvioi tuolloin, että ihmiset viettivät tablet-julkaisun parissa jopa

80 minuuttia, mikä on kaksi kertaa pidempi aika kuin tavallisesti verkkolehden parissa. [Harju et al., 2011]

Vaikka sähköisten lehtien myynti alkoi taittua syksyllä 2010, niin merkit tulevaisuuden suunnasta oli siitä huolimatta jo annettu. Lisää vauhtia kehityskululle saatiin, kun Apple julkisti tammikuussa 2011 uuden tilausmallin, joka mahdollisti kesto- ja määräaikaistilausten tekemisen. Yksi ensimmäisiä tähän tarttuneita lehtiä oli Popular Science, jonka sähköinen versio sai kuuden ensimmäisen viikon aikana yli 10 000 tilausta. The Times ja Sunday Times ilmoittivat kesäkuussa 2011, että niillä oli yli 100 000 sähköisten lehtien tilaajaa, mikä oli 28% edellistä helmikuuta enemmän. Lisäksi The Timesia ladattiin kesäkuussa 2011 keskimäärin 35 000 kertaa päivittäin. Määrä oli 40% suurempi kuin edellisessä helmikuussa. [Harju et al., 2011]

Suomen ensimmäiset tablet-lehdet nähtiin Otavamedian Suomen Kuvalehdeltä ja Suosikilta. Otavamedia kertoo sijoittaneensa tuotekehitykseen suurin piirtein saman verran kuin uuden aikakauslehden lanseeraamiseen, eli joitakin satoja tuhansia euroja. Sanoman investoinnit sen omiin julkaisuihin on arvioitu suunnilleen saman suuruisiksi. Vuoden 2011 loppuun mennessä suomalaiset lehtitalot ovat kehittäneet etupäässä iPad-julkaisuja. Seuraavana ovat vuorossa lehtien Android-versiot. [Harju *et al.*, 2011; Kettunen, 2011]

Sähköisten lehtien lukeminen on kuitenkin ollut Suomessa vielä vähäistä. Kansallisen mediatutkimuksen [2011] mukaan vain muutama prosentti suomalaisista oli syksyn 2010 ja kevään 2011 välisenä aikana käyttänyt lukulaitetta sanoma- tai aikakauslehtien lukemiseen (ks. kuva 17). Viestinnän keskusliiton projektijohtaja Kristiina Markkula uskoo laitekannan kasvun vauhdittavan lehtien lukemista. Samaa mieltä on Vihreän Langan päätoimittaja Juha Honkonen [2011], joka uskoo kriittisen massan löytävän sähköiset lehdet sitten, kun laitteet maksavat noin sata euroa ja lehdissä on jotain uutta verrattuna netti- ja paperilehtiin.



Kuva 17: Vuosien 2010-2011 taitteessa vain harva suomalainen käytti sähköistä lukulaitetta lehtien lukemiseen [Kansallinen mediatutkimus, 2011].

Mediatalouden asiantuntija Robert Picars uskoo, että muutaman vuoden kuluttua lehdet saattavat jakaa tabletteja lukijoilleen ilmaiseksi säästääkseen paperissa ja jakelukustannuksissa [Vehkoo, 2011]. Picars tietää mistä puhuu, sillä mallia on jo ehditty testaamaan Suomessa. Keväällä 2011 Aftonbladet jakoi ahvenmaalaisille tilaajilleen edullisia kiinalaisia Hanvon B10-tabletteja, joille oli mahdollista ladata päivän lehti heti sen ilmestyessä aamuyöllä. Jakelukustannuksista johtuen painettu aamulehti jaetaan ahvenanmaalaisiin koteihin vasta päivällä ja sunnuntain lehti vasta maanantaina. Pelkät paino- ja jakelukustannukset ahvenanmaalaista tilausta kohti ylittävät tilaushinnan noin kahdella kymmenellä eurolla vuodessa [Virranta, 2011a]. Ahvenanmaalaiset lukijat olivat tyytyväisiä siihen, että lehti oli kerrankin käytettävissä jo aamulla. Lisäksi sähköinen lehti sai kiitosta muun muassa helppolukuisuudesta. Kokeilu yllätti monet tahot sillä, että useat lukijat kokivat näköislehden olevan hyvä vaihtoehto sähköiseksi lehdeksi. [Virranta, 2011b]

Vaikkei jokaiselta suomalaiselta vielä löydykään tablettia, niin lukutottumuksissa on jo havaittavissa selviä muutoksia. Lokakuun puolivälissä noin 35 000 käyttäjää oli ladannut itselleen Helsingin Sanomien iPad-sovelluksen. Vielä toukokuussa vastaava luku oli noin 20 000. Tammikuussa latauksia oli kertynyt 12 000. Viikoittain kyseistä sovellusta käyttää nyt yli 10 000 käyttäjää. Helsingin Sanomat onkin ollut edelläkävijä suomalaisten sanomalehtien joukossa. Suomalaisten sähköisten sanomalahtien tarjonta on kuitenkin monipuolistumassa ja

esimerkiksi Aamulehti julkaisi iPad-sovelluksensa marraskuussa 2011. [Helsingin Sanomat, 2011; Malin, 2011]

Suomalaisten tablet-lehtien on odotettu monipuolistuvan vuoden 2011 aikana. Aikakauslehdet ovat ainakin toistaiseksi olleet sanomalehtiä aktiivisempia sähköisten lehtien kehittäjiä. Tabletin erityispiirteitä ei kuitenkaan ole vielä osattu hyödyntää kovin tehokkaasti – moni kotimainen kustantaja on lähtenyt liikkeelle pdf-näköislehdellä, joka ei kuitenkaan kehittyessään ole välttämättä huono ratkaisu [Kivioja, 2011]. Pdf, portable document format, on Adoben kehittämä sähköinen julkaisumuoto, jonka ulkoasu säilyy kaikissa käyttöjärjestelmissä samanlaisena. Media 2011 -tapahtumassa puhunut Sanoma Median digitaalisen median johtaja Michiel Buitelaar kertoi, että Hollannissa pdf-lehdet kiinnostavat lukijoita ja niiden lukukokemus on parempi iPadissa kuin verkossa. Kuitenkin jos iPad-lehti on printin kopio, jäävät tabletin tarjoamat ominaisuudet käyttämättä. Tällä hetkellä näyttää siltä että tablettijournalismin soihdunkantajia ovat ennen kaikkea aikakauslehdet. [Kivioja, 2011]

6.2 Graafinen suunnittelu ja navigointi

Vehkoon [2011] mukaan lehtien luotava itsensä uudestaan, selviytyäkseen uuden murroksen tuomista haasteista. Aiemmin printtiin liittyvät uudistukset ovat olleet lähinnä painoteknisiä ja työtapoihin liittyviä muutoksia. Nyt paine tulee ensimmäistä kertaa kuluttajilta, sillä lukulaitteiden yleistyessä tarve hyvin tehtyyn sähköiseen lehteen kasvaa nopeasti [Ruulio, 2011b].

Sähköisten lehtien ulkoasu ei ainakaan vielä ole muodostunut yhtenäiseksi. Kustantajat ovat joutuneet tekemään nopeita päätöksiä siitä, miten sisältö esitetään uudella alustalla. Uudet toimitusjärjestelmät ovatkin haasteiden edessä: niiltä odotetaan nopeutta, tehokkuutta, helppoutta ja luotettavuutta. Lisäksi alustojen määrä luo haasteita – lehden pitää taipua printtiin, tabletteihin, älypuhelimiin sekä perinteisten tietokoneiden näytöille. [Ruulio, 2011b] Haastavuutta lisää se, että parhaat käytännöt hakevat vielä itseään, eikä kukaan vielä tiedä miten tabletit lopulta vaikuttavat journalismin kuluttamiseen [Harju et al., 2011].

Uusi teknologinen lähestymistapa on tuonut markkinoille monenlaisia lehtisovelluksia, joista yksinkertaisimmat ovat lähinnä näköislehtiä ja monimutkaisimmat sisältävät runsaasti interaktiivisia elementtejä. Huomattavaa on myös, että monet päivittäiset sanomalehdet ovat omaksuneet uuden taittomallin tabletteihin, joka ei ole tabloid tai broadsheet, vaan muistuttaa enemmän aikakauslehtiä. [Heikkilä, 2011]

Kerronta ei ole myöskään aina ole lineaarista, toisin kuin aikakauslehdissä yleensä. Sähköisten lehtien suunnittelijoiden tulee ottaa huomioon erilaiset lu-

kutottumukset – lukijat haluavat todennäköisesti hypätä pois sivulta eri kohdissa. Koska tablet-lehdet ovat ennen kaikkea visuaalisia, niin suunnittelijoiden ja kuvaajien on hyvä olla mukana suunnitteluprosessissa alusta alkaen. Tablet-ympäristö edellyttää myös perinteistä verkkojournalismia suurempaa yhteistyötä multimedian tuotannossa ja suunnittelussa. Tarjolla on runsaasti interaktiivisten elementtien käyttömahdollisuuksia ja erilaisia esitysmuotoja, jolloin kokonaisuus voi huonolla suunnittelulla muodostua kaoottiseksi. Siksi onkin tärkeää rakentaa tarina ja ulkoasu lukijan kokemuksen ympärille. [Harju *et al.*, 2011]

6.2.1 Julkaisutyypit

Harjun [*et al.,* 2011] mukaan julkaisujen perustyypit voidaan karkeasti jakaa neljään kategoriaan, jotka ovat pdf, verkko, tablet ja multimedia.

Pelkistetyimmillään sähköiset lehdet ovat näköislehtiä, joissa ei juuri ole interaktiivisuutta tai muita toimintoja. pdf-tyypillä tarkoitetaan, että julkaisu koostuu pelkästään paperilehdelle identtisestä näköislehdestä (ks. kuva 18). Jotkut sähköiset lehdet ovat pelkästään pdf-sivuista koottuja näköislehtiä, joissakin pdf-sivut ovat valinnainen toiminto, jolla voi tarkastella näköislehteä. Etusivu on niin ikään toisinto paperiversiosta, eikä siitä ole linkkejä lehden juttuihin tai osastoihin. Tällöin myös lehden sisällöllä on tapana edetä lineaarisesti noudattaen paperilehden sivu-metaforaa. Askeleen pidemmälle mennään julkaisuissa, joissa etusivut ovat kansikuvia. Tällöin etusivu on identtinen paperiversion kanssa, mutta muutoin lehteä on muokattua ainakin osittain tablettia varten. [Harju *et al.,* 2011]

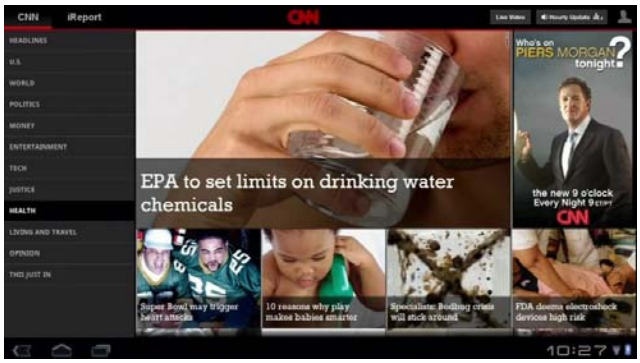
Kuva 18: Iltasanomien tablet-julkaisu on pdf-typpinen näköislehti.

Harjun [*et al.,* 2011] mukaan verkko-julkaisulla tarkoitetaan sitä, ettei juttujen rakenne juuri hyödynnä tabletin ominaisuuksia, vaan muistuttaa enemmän perinteistä verkkosivulla olevaa juttua (ks. kuva 19). Joissakin tablet-

julkaisuissa tämä on vallitseva piirre, kun taas toisissa on rinnakkain paljon verkkosivunkaltaista aineistoa ja tabletille optimoitua sisältöä.

Kuva 19: Financial Times on verkko-tyyppinen julkaisu.

Tablet-julkaisut puolestaan sisältävät runsaasti tabletin sormella ohjattavalle käyttöliittymälle optimoitua aineistoa. Julkaisujen ulkoasu on monesti verkkoa pelkistetympi ja kuvat ovat hallitsevammassa roolissa (ks. kuva 20). Ulkoasussa ja navigoinnissa on pyritty huomioimaan, että lehteä selataan sormella, eikä tarkalla hiirellä. Koska tabletin näyttö on olennaisesti painettua julkaisua pienempi, esitetään yhdellä sivulla yleensä vähemmän aineistoa. Monet tablet-julkaisut käyttävätkin tilan oivallisesti hyväksi. On tyypillistä, että kehittyneessä tablet-julkaisussa yksittäistä kuvaa koskettamalla saa näkyviin kuvagallerian tai liikkuvaa kuvaa – joissakin julkaisuissa kuvan esineitä voi jopa pyörittää ja katsoa haluamastaan kulmasta. Multimedia-julkaisut sisältävät vielä tablet-julkaisujakin enemmän erilaisia interaktiivisia sisältöjä ja näyttäviä grafiikoita, jotka on optimoitu tabletille (ks. kuva 21). [Harju, et al., 2011]



Kuva 20: CNN:n tablet-tyyppisessä julkaisussa kuvat ovat isossa roolissa.

Kuva 21: Wired on tabletille optimoitu multimedia-tyyppinen julkaisu.

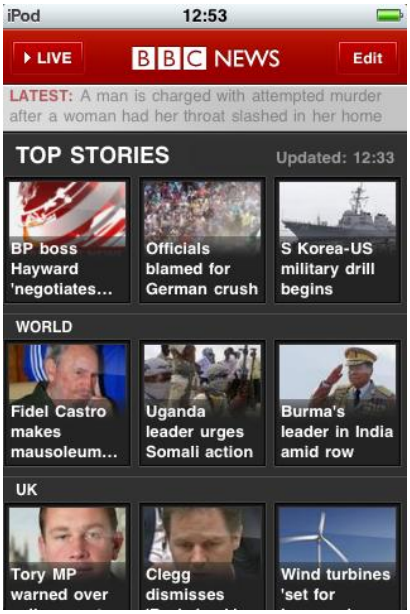
6.3.2 Navigointi

Tablet-julkaisujen sisällön ja osastojen välisissä navigointitavoissa on myös suurta vaihtelua. Painikkeiden ja säätimien paikat hakevat vielä lopullista sijoituspaikkaansa – julkaisusta riippuen ne saattavat sijaita ruudun millä tahansa reunalla tai kulmissa. Tulevaisuus näyttää, mitkä navigointitavat osoittautuvat käyttäjäystävällisimmiksi ja jäävät elämään. Seuraavaksi luodaan lyhyt katsaus tällä hetkellä käytössä oleviin navigointitapoihin.

Pdf-navigointi tarkoittaa sitä, että julkaisu on käytännössä identtinen painetun lehden kanssa. Navigointi tabletilla tapahtuu sormea liikuttamalla, eikä linkkejä yksittäisiin juttuihin tai erillisiin osastoihin ole. Pdf-nauha eroaa hieman tavallisesta pdf-versiosta. Se tarkoittaa, että sivut muistuttavat pdf-sivuja, mutta sivuttain kulkevan horisontaalisen järjestyksen lisäksi yksittäiset jutut ja osastot on koottu myös vertikaalisesti päällekkäin.

Pudotusvalikko on yksi navigoinnin perustavoista. Se tarkoittaa, että jollakin sivulla olevasta painikkeesta saadaan avattua tekstitaulukko josta on linkit osastoihin. Monissa tablet-julkaisuissa se on yksi navigointivaihtoehto muiden rinnalla. [Harju *et al.,* 2011]

Liukumatriisin tai matriisin avulla navigoiminen tarkoittaa puolestaan sitä, että sivu on jaettu vaaka- ja pystysuuntaisiin sormella selattaviin lohkoihin (ks. kuva 22). Ne kaikki edustavat jotain lehden osastoista. Valitsemalla jonkun lohkoista käyttäjä pääsee käsiksi valitsemaansa juttuun ja saa näkyviin lisää lohkoja samasta aihepiiristä. Matriisi-tyyppinen etusivu on niin ikään koottu erikokoisista lohkoista, joilta voi siirtyä esimerkiksi tärkeimpiin uutisaiheisiin. [Harju, et al., 2011]



Kuva 22: BBC News hyödyntää matriisi-tyyppistä navigointia.

Kuvatarjotin muistuttaa paperilehden sisällysluetteloa, josta voi siirtyä haluamaansa osastoon tai juttuun. Se ei ole yleensä koskaan ainoa navigointitapa, vaan vaihtoehto muiden rinnalla. Karuselliksi kutsuttu navigointitapa pyrkii hyödyntämään tabletin käyttöliittymää. Esimerkiksi The Daily antaa käyttäjälle ensimmäisenä näkymänä osastonavigoinnin karusellimaisena näkymänä, jota voi pyöritellä sormella. Karuselli-tyyppinen etusivu tähtää niin ikään tabletin käyttöliittymän hyödyntämiseen. ABC News on tehnyt uutta ajattelua edustavan ratkaisun ja jaotellut uutiset kolmiulotteisen pallon pinnalle, jota lukija voi sormella pyöritellä kaikkiin suuntiin (ks. kuva 23). [Harju, *et al.*, 2011]

Kuva 23: ABC Newsin etusivu pyrkii hyödyntämään tabletin ominaisuuksia. Navigoinnissa käytetään kolmiulotteista palloa, jota liikutellaan sormella.

Varsin yleinen navigointitapa on tällä hetkellä kuvanauha, johon on upotettu juttu- ja osastolinkkejä. Selattavalle nauhalle on kronologisesti sijoitettu juttuihin ja osastoihin lyhyesti viittaavia kuvia ja lyhyitä tekstipätkiä (ks. kuva 24). Liukunauha puolestaan tarkoittaa niin ikään sormella selattavaa nauhamaista luetteloa, jossa on yleensä osastojen nimiä. Siinä ei kuitenkaan ole kuvia kuvanauhan tapaan. Myös liukunauha on monesti yksi navigointivaihtoehto muiden rinnalla. [Harju, et al., 2011]

Kuva 24: Time-lehti hyödyntää tablet-julkaisussaan alalaidassa kulkevaa kuvanauhaa. Etusivu puolestaan on painetun lehden kaltainen.

Kaikkien eri navigointitapojen kategorisointi on vielä hankalaa, sillä tablet-julkaisut elävät vielä ensivaihettaan. Tablet-julkaisujen tarkastelu osoittaakin, että niiden etusivuissa, navigoinnissa ja julkaisutavoissa on suurta vaihtelua (ks. kuva 25). Muina navigointitapoina voidaan vielä pitää ainakin kioskia, joka tarkoittaa sitä, että lehti on pilkottu omiin erikseen ladattaviin osastoihin. Osastot ovat itsenäisiä, tablet-lehtien kaltaisia julkaisuja. Paras esimerkki tästä on The Sunday Times, jonka osastot tulee ladata erikseen. Etusivun muiksi tyypeiksi voidaan lukea vielä esimerkiksi kuvatarjotin, joka koostuu erikokoisista kuvista ja niihin upotetuista tekstipätkistä, live-sivu, joka perustuu liikkuvaan kuvan ja multimediatarjotin, joka viittaa etusivuun, johon on upotettu erilaisia tabletille optimoituja multimediaelementtejä. [Harju, *et al.*, 2011]

Julkaisu-app	Etusivun rakenne									Osastojen navigointi									Juttutyypit			
	PDF	Kansikuva	Matriisi	Tablet	Liukumatriisi	Karuselli	Kuvatarjotin	Live-sivu	Multimediatarjotin	PDF	PDF-nauha	Pudotusvalikko	Liukunauha	Kuvatarjotin	Karuselli	Kuvanauha juttulinkein	Liukumatriisi	Kioski	PDF	Verkko	Tablet	Multimedia
ABC				2		1						1				2				x		
Associated Press									x			x	x							x		
BBC					x												x			x	x	
Bild HD									x						x							x
CNN							x					1				3	2				x	
DN+		x									x					II				x		
Economist		x														x				x		
Esquire								x								x					x	
Financial Times				x									x							x		
Frankfurter Rundschau				x								x				x					x	
Helsingin Sanomat			x								x									x		
Iconist							x							1		2						x
Iltalehti	x									x									x			
Iltasanomat	x									x									x			
L.A.Times				x								x								x	x	
L.A.Times Magazine		x									x					x			x		x	
Le Monde				x									x							x	x	
Net-A-Porter		x												x		x					x	
Newsweek	x									x									x			
NPR					x												x			x		
PAIS				x								x									x	
Project								x								1						x
Pulse					x												x			x		
San Francisco Chronicle			x												x					x	x	
Slate					x							x				x				x		
Suomen Kuvalehti		x												x		x				x		
The Age							x					x									x	
The Daily						x							x		x	x						x
The Sunday Mor. Herald							x					x									x	
The Sunday Times		x														2		1			x	
Time		x										x				x	x			x		
Times				x									x			x				x		
USA Today				x								x								x		
VG+		x										x				x				x	x	
Washington Post			x									x								x	x	
Wired		x									x					II						x

Kuva 25: Sähköisten lehtien ulkoasu, navigointitapa ja etusivun rakenne vaihtelevat suuresti eri julkaisujen välillä [Harju *et al.*, 2011].

Yhteistä lähes kaikille tablet-julkaisuille on kuitenkin se, että muutamat tavallisimmat kosketuseleet toimivat niissä samalla tavalla. Sivujen välillä siirtyminen, zoomaus, vierittäminen ja kohteen valinta ovat jo melko vakiintuneita komentoja (ks. kuva 26).

Kuva 26: Tavallisimpia sähköisten lehtien lukemisessa käytettäviä eleitä.

On vielä liian varhaista sanoa, mitkä ulkoasu- ja navigointimallit jäävät elämään. Tabletin tarjoamien ominaisuuksien hyödyntäminen nähdään sen sijaan jo nyt hyvin oleellisena sähköisien lehtien kokonaisuuden suunnittelussa. Mario Garcia kehottaakin unohtamaan ajatuksen siitä, että lehdet siirrettäisiin sellaisenaan tableteille. Itse asiassa hän kehottaa tyystin unohtamaan painetun lehden olemassaolon ja keskittymään uuden julkaisumallin luomiseen. ”Tabletin pitää tehdä sormi onnelliseksi”, hän toteaa. Lisäksi Garcia suosittelee, että sähköisiä lehtiä suunniteltaessa käytettäisiin kaikkia osaajia – tarinankertojia, graafikkoja, päätoimittajia, teknologia-osaajia ja markkinoijia. [Harju *et al.*, 2011]

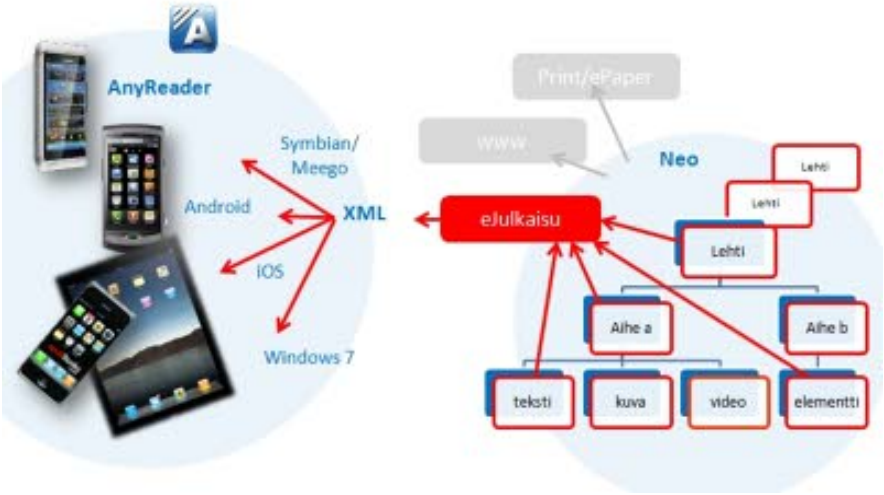
6.3 Valmiit julkaisualustat

Koska erilaisia laitteita on paljon, aiheuttavat niiden eri käyttöjärjestelmät ja tekniset vaatimukset päänvaivaa julkaisijoille. Tablettiversio aikakauslehdestä tehdään yleensä olemassa olevan taiton pohjalta. Sen pitää kuitenkin istua useaan eri formaattiin, joiden koot vaihtelevat. Monesti luodaan lehdestä sekä pysty- että vaakataitto, jotka sopeutuvat tabletin asentoon – jos laite on kyljellään, niin näytetään vaakataitto, jos se on pystyssä, niin esillä on pystytaitto. Sen jälkeen kuvaan astuvat interaktiiviset elementit, valokuvat, video ja animaatiot – sähköisen lehden etu on, että se voi sisältää painetun lehden ominaisuuksien lisäksi lähes mitä tahansa. [Kinturi, 2011a]

Erityyppisille laitteille muokkaamisessa on käytössä muutamia perusperiaatetta. Tavallisin on xml-pohjaisiin sisältövirtoihin pohjautuva ohjelmisto, jolla sisältö mukautetaan automaattisesti kaikille laitteille. Paras lopputulos kuiten-

kin syntyy kullekin laitteelle suunnitellusta ja taitetusta julkaisusta. Tähän tarvitaan työkaluja eli ohjelmistoja, jotka ottavat huomioon laitteen vaatimukset. Jälkimmäinen vaihtoehto on kuitenkin työläämpi ja kalliimpi. [Kinturi, 2011]

Uudet julkaisualustat ovat tuoneet markkinoille niille suunnattuja työkaluja. Eräs tällainen median sisällön suunnittelu- ja hallintajärjestelmä on sanomalehtipuolen ohjelmistoihin erikoistuneen Anygraafin Neo Publishing Solution (ks. kuva 27). Samaisen yhtiön AnyReader puolestaan tarjoaa helpon, dynaamisen ja älykkään julkaisutavan eri kanaville, kuten tableteille. Järjestelmä perustuu siihen, että sama xml-paketti lähetetään kaikille laitteille, joissa se tuottaa niille sopivan ulkoasun (ks. kuva 28). Älykkäät ja mukautuvat taittopohjat mahdollistavat sen, ettei kustantajien tarvitse tehdä erilaisia ulkoasuja jokaiselle laitteelle erikseen. [Anygraaf, 2011]



Kuva 27: Neo Publishing Solution on median sisällön suunnittelu- ja hallintajärjestelmä, joka tarjoaa muun muassa tehokkaat julkaisukeinot eri kanaviin.

Kuva 28: Anygraafin AnyReader tuottaa automaattisesti kullekin laitteelle sopivan ulkoasun ja resoluution.

Myös hollantilainen WoodWing-ohjelmisto vie mobiilisisältöjä erilaisiin päätelaitteisiin. Kyseisessä järjestelmässä sisältöä voidaan viedä tableteille ja päätelaitteille kolmella eri mallilla. Ensimmäinen niistä on ”sanomalehtimalli”, jossa laite lataa sisällön verkosta samaan tapaan kuin tietokoneen selaimella. Toinen keino on, että päätelaite itse luo julkaisun tiettyjen sääntöjen perusteella – siis samaan tapaan kuin AnyReaderin tapauksessa. Kolmas vaihtoehto on, että julkaisu taitetaan erikseen kyseiseen päätelaitteeseen. Palvelinpohjainen WoodWing on viimeksi mainitussa toteutustavassa mainio, sillä graafinen suunnittelija voi ohjelmistolla tarkistaa, miltä julkaisu näyttää ja muokata sitä tarpeen mukaan paremmaksi. [Kinturi, 2011a]

Suuri määrä sähköisten lehtien lukemisesta tapahtuu lehtien omien ladattavien sovellusten avulla. Internetiin on syntynyt myös useita lehtikauppoja, jotka toimivat samalla julkaisualustoina. Ilmaisten sovellusten avulla voi ostaa ja lukea lukuisia lehtiä. Valikoimasta löytyy jo tuhansia sähköisiä sanoma- ja aikakauslehtiä ympäri maailmaa. Tarjolla on myös useita suomalaisia lehtiä. Yksi suurimmista sähköisistä lehtikioskeista tableteille on Zinio, joka julkaisi oman sovelluksensa pian iPadin julkistamisen jälkeen (ks. kuva 29). Myöhemmin Zinio tuotiin myös Android-laitteille. Sovellus on käyttäjille ilmainen. Se sisältää valtavan määrän lehtiä ympäri maailmaa, joita käyttäjä voi halutessaan ostaa ja ladata tabletille. [Dodge, 2010] Vastaavan tyylinen suomalainen sovellus on lehtiluukku, jonka sovellus on toistaiseksi saatavilla vain Applen laitteille. Laaja lehtivalikoima löytyy myös sanomalehtiin keskittyneeltä Press Readerilta. Zinio, Lehtiluukku ja Press Reader ovat muutamia esimerkkejä markkinoille ilmestyneistä uudenlaisista lehtipisteistä. Yhteistä niille kaikille on se, että lehdet ovat aluksi olleet lähinnä näköislehtiä, joissa ei juuri ole hyödynnetty tablettien tarjoamia ominaisuuksia. Tilanne on kuitenkin pikku hiljaa muuttumassa ja lehdet ovat menossa kohti interaktiivisempaa lukukokemusta.

Kuva 29: Zinion sähköinen lehtikioski sisältää tuhansia lehtiä.

Ahlrothin [2011] mukaan erilliset tablet-sovellukset ovat aluksi olleet lähinnä suurten lehtien etuoikeus, mutta tilanne on muuttumassa. Kun sovellusten tekoa varten kehitetään avoimen lähdekoodin ilmaisia sovelluksia, niin pientenkin firmojen mukaan lähteminen helpottuu ja riskit pienenevät merkittävästi kulujen myötä. Lisäksi Lehtiluukun kaltaiset järjestelmät tarjoavat pienillekin lehdille helpon ja nopean väylän sähköisen lehden julkaisemiseen. Honkonen [2011] kertoo, että Vihreän Langan iPadille vieminen vei kyseisen sovelluksen kautta vain muutamia päiviä.

6.4 Hinnoittelu ja mainonta

Mediataloissa tabletit ovat herättäneet kiinnostusta, vaikka alkuinnostuksen jälkeen ollaankin huomattu, että tuotekehitykseen kuluu aikaa ja rahaa. Samalla on kuitenkin huomattu, että ihmiset ovat valmiita maksamaan mobiili- ja tablet-sovelluksista [Vehkoo, 2011]. Aiheen takana piilevä innostus juontaa juurensa siitä, että verkkouutisten ilmaisuutta on myöhemmin ajateltu virheenä, jota ei enää saada tekemättömäksi. Tablettien ja mobiililaitteiden kautta onkin nyt huomattu uusi väylä, joka voisi viedä takaisin maksulliseen sisältöön. [Vehkoo, 2011]

Lehtien on paneuduttava myös kustannusstrategiaan. Sisältöä voidaan jakaa ilmaiseksi tai siitä voidaan ottaa maksu. Jos maksu otetaan, niin se voidaan hoitaa tilausmaksuna, artikkelikohtaisesti, jatkuvana tilauksena ja niin edelleen. Vaihtoehtoja on paljon, eikä parhaita keinoja vielä tiedetä. Oman mausteensa soppaan tuovat kolmannet tahot, jotka ottavat tilauksista siivun. Esimerkiksi Apple on vaatinut 30 prosentin siivun Apple Storen kautta kulkevista tilauksista. Lisäksi tilaajatiedot jäävät ainakin tällä hetkellä vain Applen tietoon. [Kinturi, 2011] Tilanne on joidenkin mielestä kestämätön. Esimerkiksi Matkalehden päätoimittaja Jorma Aula toteaa, ettei iPadiin kannata lähteä mukaan, koska se on Applelle pelkkä rahastuskeino. Matkalehti hyppää mukaan vasta, kun lukulaitteet tarjoavat lehdelle parempia ansaintamahdollisuuksia. [Kinturi, 2011b]

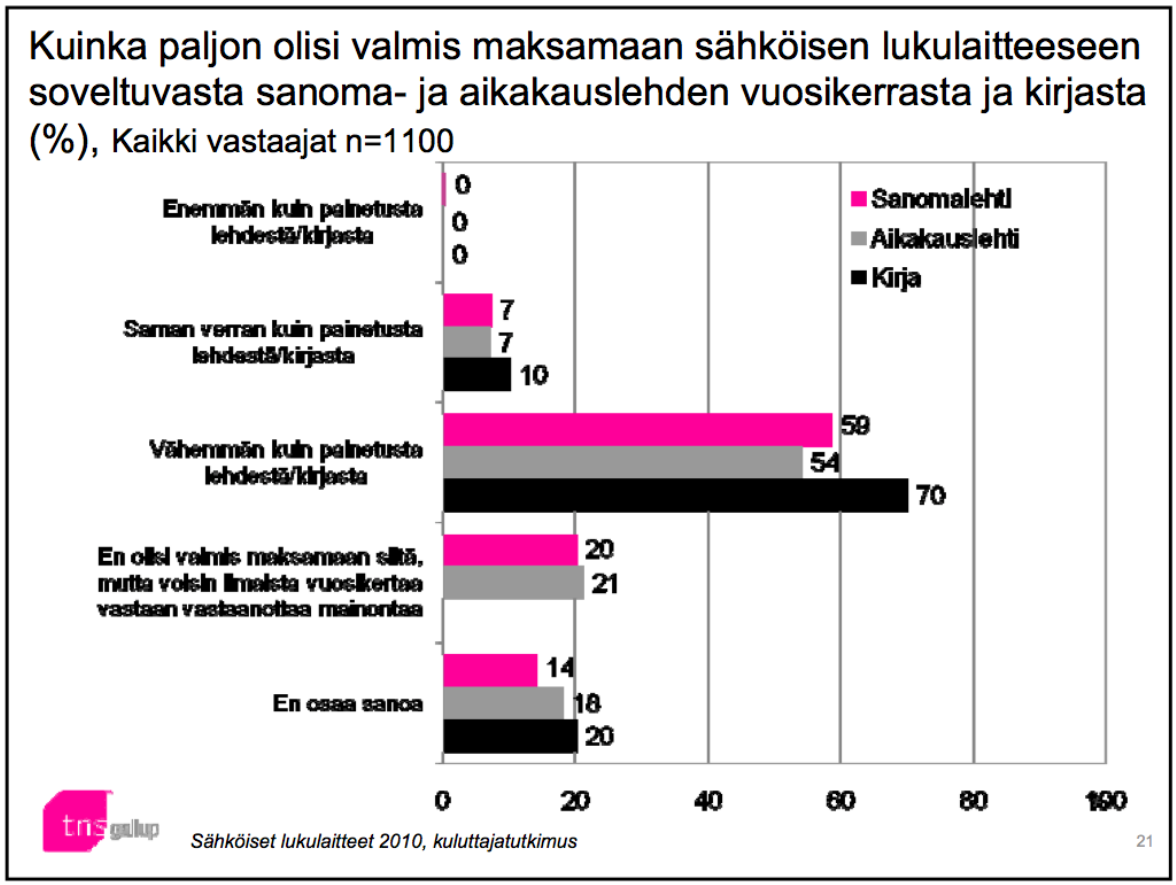
Monen muun tabletteihin liittyvän asian ohella myös hinnoittelupolitiikka on vielä epäselvä ja epäyhtenäinen. Irtonumeroiden hinnat vaihtelevat runsaasti: osa julkaisijoista hinnoittele sähköisen version samanhintaiseksi painetun lehden kanssa, osa edullisemmaksi ja osa on jakanut sähköistä versiota jopa ilmaiseksi saadakseen ihmisiä tutustumaan uuteen tuotteeseen.

Suosikki-lehti oli ensimmäisiä suomalaisia aikakauslehtiä, joka teki oman versionsa tabletille. Lokakuussa 2010 julkaistu sähköinen lehti kuitenkin haudattiin nopeasti. Syinä huonoon menekkiin pidettiin tablettien vähäisyyttä julkaisuajankohtana ja yhtenäistä hinnoittelua printtiversion kanssa. Lehden päätoimittaja Ville Kormilainen piti kokeilua kuitenkin hyvänä, sillä se osoitti, että

julkaiseminen uudella alustalla onnistuu. Lisäksi lehdessä huomattiin, että toimitus ja avustajat alkoivat oma-aloitteisesti antaa uuden julkaisunmuodon parantamiseen tähtääviä ideoita. [Virranta, 2011c]

Suosikin tapaus nostaa kuitenkin esiin ongelman, joka lehtien tulee ratkaista: miten hinnoitella lehden sähköinen versio? Vihreän Langan päätoimittaja Juha Honkonen [2011] pitää järkevänä hinnoitella sähköisen lehden printtiversiota halvemmaksi, sillä kulutkin ovat pienemmät – jakelu- ja painokustannuksia ei käytännössä ole. Vihreä Lanka kokeilee tällä hetkellä ilmaisjakelua vuoden 2011 lokakuussa julkaistulle iPad-lehdelleen. Tarkoituksena on tehdä lehti houkuttelevammaksi ja saada uusia lukijoita. Sähköinen versio on tarkoitus muuttaa maksulliseksi ensi vuonna.

Myös yleinen mielipide tuntuu vieroksuvan sitä, että sähköinen lehti olisi painettua kalliimpi. Kansallisen mediatutkimuksen [2011] tekemän kuluttajatutkimuksen mukaan valtaosa vastaajista vaatii edullisempaa sähköistä versiota (ks. kuva 30). Kukaan vastaajista ei olisi valmis maksamaan sähköisestä versiosta printtiä enempää.

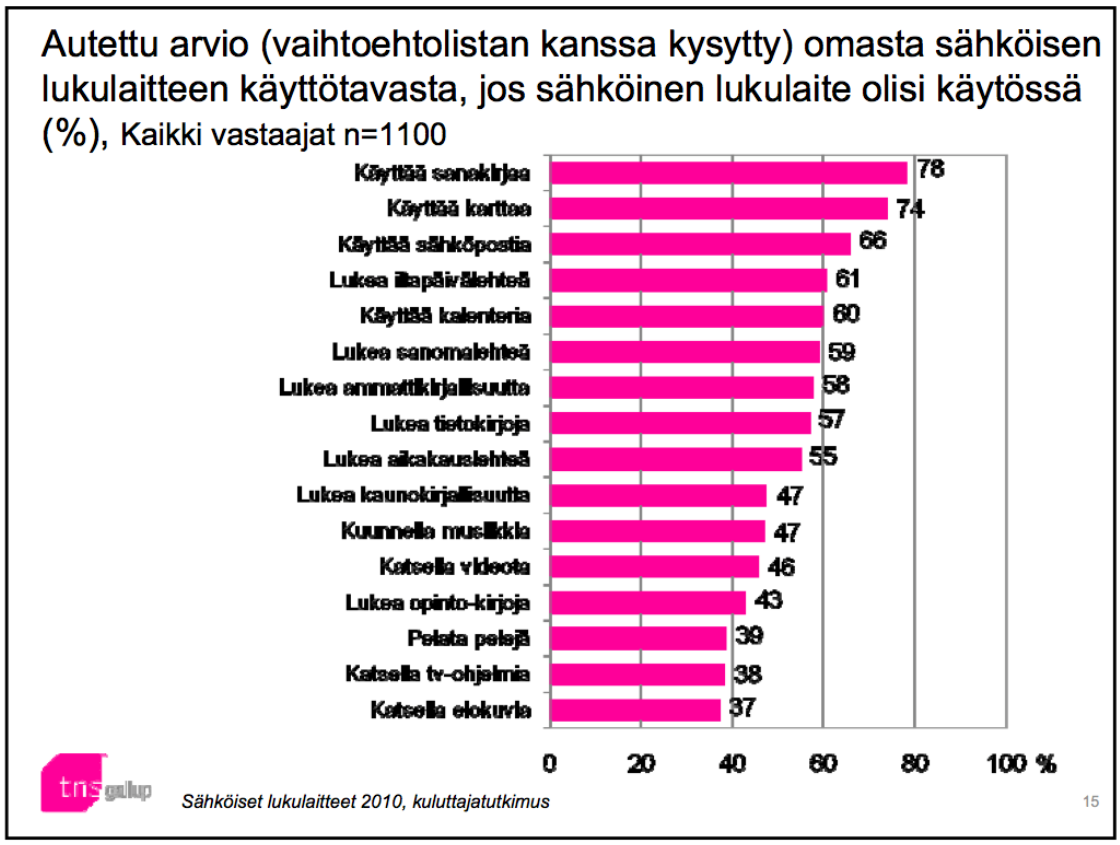


Kuva 30: Sähköisestä lehdestä halutaan maksaa vähemmän kuin painetusta lehdestä. [Kansallinen mediatutkimus, 2011]

Tablettien sisällöt ovat myös erittäin houkutteleva mainosympäristö. Kiinnostava yksityiskohta mediataloille onkin se, että kuluttajien ensikokemusten perusteella mainonta on nähty hyväksyttävänä ja sisältöä tukevana osa-alueena. Lisäksi moni sanoo olevansa valmis vastaanottamaan mainontaa lehden maksutonta vuosikertaa vastaan (ks. kuva 30) [Kansallinen mediatutkimus, 2011]. Sähköisissä lehdissä mainostaminen on kuitenkin hyvä tehdä lukukokemuksen ehdoilla [Malin, 2011]. Tableteissa mainostaminen voi viedä lehtimainonnan uudelle tasolle, sillä kiinnostavien mainoksien kautta lukija voi helposti siirtyä mainostajan sivuille – sillä edellytyksellä, että lehdessä on käytetty toiminnallisuuksia, joiden kautta siirtyminen käy mainosta painamalla.

6.5 Sähköinen tulevaisuus

Tablettien ja sähköisten lehtien markkinat ovat herkässä kehitystilassa. Tällä hetkellä tabletit vaikuttavat olevan ennen kaikkea aamulla ja illalla käytettäviä mediankulutusvälineitä, mikä vaikuttaa lehtitalojen näkökulmasta katsottuna positiiviselta ilmiöltä (ks. kuva 31) [Heikkilä, 2011]. Yksi suurimmista oivalluksista tablettien kohdalla vaikuttaakin olevan se, että yksi esine sisältää monta mediaa – mahdollisesti jopa kaikki ne, joita käyttäjä haluaa seurata. Laitteen ruudulla journalismin rinnalla sijaitsee koko Internet, pelit, kartat, kirjat ja sosiaalisen median vuorovaikutus sekä käyttäjän ulottuvilla olevat tuhannet sovellukset [Harju *et al.,* 2011]. On täysin mahdollista, että tabletit tulevat huomaamattomasti lipumaan osaksi ihmisten arkipäivää, hieman samaan tapaan, kuin kännykät 1990-luvun aikana. Lehtitalojen näkökulmasta katsottuna, markkinoinnissa saattaisikin olla hyödyllistä korostaa tablettia ”lehdenlukualustana, jolla voi tehdä kaikkea”. Itse lehteä ajatellen, saattavat helppokäyttöisyyden ja uudenlaisen interaktiivisen lukukokemuksen korostaminen olla niitä tekijöitä, joilla herätetään kuluttajien kiinnostus. Positiivista sähköisten lehtien nykytilanteessa on myös se, että ihmiset käyttävät huomattavasti enemmän aikaa niiden lukemiseen verrattuna lehtien verkkosivuihin [Heikkilä, 2011].



Kuva 31: Tabletit vaikuttavat olevan ennen kaikkea mediankulutusvälineitä [Heikkilä, 2011.] Myös Kansallisen mediatutkimuksen [2011] tekemä tutkimus puoltaa tätä näkökulmaa.

Lienee jo nyt täysin selvää, että tablettien ominaisuudet tarjoavat mahdollisuuksia tulevaisuuden journalismille. Tablettien journalismi saattaakin tulevaisuudessa sisältää monia mahdollisia muotoja. Toiset julkaisut saattavat panostaa visuaalisuuteen lisäten kuvia ja televisiokerrontaa, samalla kun toiset painottavat datajournalismia ja vuorovaikutteista grafiikkaa [Harju *et al.*, 2011]. Ahlroth [2011] ennustaakin, että "kanavapakettibisnes" on valtaamassa koko kustantamisen kentän. Se tarkoittaa, että mediakonsernit alkavat myydä eri sisältömuotoja – televisio- ja radio-ohjelmia, lehtijuttuja ja muuta media – samassa paketissa. Älypuhelimista tullee ensisijainen nopean uutissisällön kulutusväline, mutta tableteilta halutaan enemmän. Laadukas sisältö yhdistettynä visuaalisiin yllätyksiin luo puitteet ennen kaikkea tableteilta operoitaville aikakauslehdille, mutta myös sanomalehdillä on kaikki mahdollisuudet tehdä siitä omansa. Huomattavaa on myös, että monet päivittäiset sanomalehdet ovat jo nyt omaksuneet uuden taittomallin sähköisiin lukulaitteisiin, joka ei muistuta perinteisiä muotoja, vaan on enemmän aikakauslehtien kaltainen. Ahlrothin mielestä tabletti on paperia jopa parempi käyttöliittymä sanomalehdille. Helposti mukana kulkeva laite käynnistyy nopeasti ja sitä on helppo käyttää missä

ja milloin tahansa. Kuluttajan kannalta on kätevää, että tilattu lehti ilmestyy laitteeseen heti julkaisuhetkellä. Lisäksi vanhat numerot säilyvät muistissa muodostaen kätevän digitaalisen juttuarkiston. [Ahlroth, 2011]

Journalismilta odotetaan tulevaisuudessa ennen kaikkea innovaatioita, joiden avulla hyödynnetään tablettien ominaisuuksia ainutlaatuisten kokonaisuuksien luomiseen, eikä vain tyydytä siirtämään jo olemassa olevaa materiaalia uudelle alustalle. Median tuottajilta vaaditaankin tulevaisuudessa uudenlaista ymmärrystä siitä, mitä journalismi on ja mitä se voi olla. Tähän tarvitaan uudistumista: on siirryttävä julkaisemisesta osallistumiseen, uutisoinnista keskusteluun ja yksilöistä joukkoälyyn. Innovaatioita ei tule nähdä pelkkänä tekniikkana, sovelluksina ja sisältöinä, vaan uudenlaisena lähestymistapana, jossa huomioidaan verkostoitumisen, digitaalisen teknologian ja uusien alustojen muutokset mediamaailmassa. [Hermida, 2011; Harju *et al.*, 2011]

Journalismin odotetaan tarttuvan uuden ympäristön tarjoamiin mahdollisuuksiin. Mediatalojen ja journalistien on löydettävä uusia tapoja tehdä journalismia, sillä pelkkä sisällön kopioiminen painetusta lehdestä ja nettisivuilta ei riittäne tablet-version sisällöksi tulevaisuuden digitaalisessa mediaympäristössä. Juttujen suunnittelussa tulee ottaa lukukokemus huomioon – oli kyse sitten kerronnasta tai navigoinnista. Kaikki viittaa siihen, että tulevaisuudessa mediatalot tarvitsevat yhä enemmän myös teknologiaosaamista ja -suunnittelua. [Harju *et al.*, 2011]

Tabletit ovat viemässä lehtien lukemista myös lukijalähtöisempään ja personoitavampaan suuntaan. Lukijalähtöisyys edellyttää taitavaa sisältöjen ja formaattien suunnittelua. Ulkoasun suunnittelijoiden ja juttujen työstäjien yhteinen näkemys saattaa olla askel oikeaan suuntaan hyvän lukukokemuksen tavoittelussa. Mediatalojen tulee ottaa kantaa myös siihen, miten työnjako lopulta tehdään ja käytetäänkö talon ulkopuolisia osapuolia. Kun lukukokemukset alkavat painaa yhä enemmän, on pohdittava, miten digitaalinen aineisto on muokattavissa vastaamaan kohderyhmän mieltymyksiä. Kun lukeminen ei välttämättä tapahdu lineaarisesti, on navigointiin ja tarinan etenemiseen kiinnitettävä entistä enemmän huomiota. Tablettien lehtosovellukset suunnitellaan ”sormille ja aivoille”, joten sekä liikkeeseen että luettavuuteen on panostettava. Navigointi tulee rakentaa helpoksi ja ymmärrettäväksi, jotta lukijan päähuomio ei karkaa lehden sisällöllisistä seikoista. Hyvän kokonaisuuden rakentaminen on haaste, sillä tabletit mahdollistavat monien toimintojen hyödyntämisen. Tablettia voi käännellä, sisältöä voi zoomata ja siirrellä, aktiivisia elementtejä voi koskettaa ja juttujen sekä julkaisujen sisällä voi liikkua monin tavoin. Samalla lukijalle tulee tarjota myös pysyvämpiä maamerkkejä, joiden avulla navigointi pysyy helppona. [Harju *et al.*, 2011]

Sisältö ja muoto tulevat luultavasti kulkemaan entistäkin vahvemmin rinnakkain. Sähköisten lehtien tuotanto saattaa alkaa muistuttaa multimediatuotantoa, jossa työprosessit kulkevat rinnakkain ja toistensa ehdoilla. Kun myös tabletit ovat käyttäjiensä personoitavissa, on mahdollista, että koko lukukokemus on matkalla huomattavasti painettua lehteä yksilöllisempään suuntaan. Tablettien ja sähköisten lehtien yhteiselo saattaakin innostaa uusia kohderyhmiä lehtien pariin – se sopii erityisesti nuorten sukupolvien liikkuvaan ja yksilöllisyyttä, kuluttajuutta sekä verkostoitumista korostavaan elämäntapaan. [Harju *et al.*, 2011]

Mediatalojen ja pienempien kustantajien on nyt aika avata silmät ja katsoa tulevaan. On turha vedota siihen, että tabletteja on ihmisillä vielä vähän, sillä tilanne saattaa muuttua äärimmäisen nopeasti – ellei ole jo muuttunut. Ei kannata myöskään olla varma, etteivät ihmiset luovu sanomalehden lukemisesta aamupalapöydässä. Kaikki eivät varmasti luovukaan, mutta kun laitteet yleistyvä ja sähköisen lehden lataaminen tulee tarpeeksi helpoksi, niin painettujen lehtien levikit kääntynevät yhä jyrkempään laskuun. Silloin vahvoilla ovat ne tahot, jotka ovat parhaiten valmistautuneet digitaaliseen tulevaisuuteen – eli ne, joilla on jo laadukasta ja lukijaystävällistä sisältöä laitteille. [Ruulio 2011a].

Mario Garcia, yksi maailman johtavista sanomalehtien ulkoasun suunnittelijoista, pitää mahdollisena, että tabletit tulevat toimimaan mediateollisuuden muutoksen keskiössä. Garcian mukaan ei ole enää kyse siitä, julkaistaanko jokin lehti tabletilla vai ei. Kyse on siitä, koska ja miten se julkaistaan. Hän kehottaakin jokaista lehteä julkaisemaan ensimmäisen tablet-versionsa mahdollisimman pian. [Harju *et al.*, 2011] Samaa mieltä on Vihreän Langan päätoimittaja Juha Honkonen [2011]. Hän uskoo, että kuluvan vuosikymmenen lopulla tableteilla ja puhelimilla luetaan Suomessa enemmän lehtiä kuin paperilta. Honkonen uskoo ensi vuosikymmenen olevan paperilehden hitaan kuoleman aikaa. Hän toteaa, että paino-, jakelu- ja verotuskustannusten noustessa ”paperilehdestä saattaa tulla ylellisyystuote samaan tapaan, kuin jotkut paksut muotilehdet ovat jo nyt”. Kun tablettien ominaisuuksia opitaan hyödyntämään paremmin, tulee sähköisiin lehtiin toimintoja, jotka ratkaisevat pelin lopullisesti niiden hyväksi.

Kaiken teknologiapaatoksen keskellä on kuitenkin hyvä muistaa, ettei hienoinkaan paperinen tai sähköinen lehti ole riittävän hyvä ilman laadukasta journalistista sisältöä. Helsingin Sanomien päätoimittaja Riikka Venäläinen uskookin, että ”ihmiset haluavat maksaa tulevaisuudessakin laadukkaasta, yllättävästä ja viihdyttävästä journalismista, oli kanava mikä hyvänsä” [Jokinen, 2011].

7. Yhteenveto ja loppupäätelmät

Sanoma- ja aikakauslehtibisnes ehti pysyä vuosikymmeniä lähes muuttumattomana. Siitä seurasi jopa tietynlainen kaavoihin kangistuminen, josta ollaan pikku hiljaa pääsemässä irti. Vaikka Internetin läpimurto on ollut lehtitaloille kirous, joka on tasaisen varmasti syönyt lehtien levikkejä, niin se saattaa loppujen lopuksi kääntyä niille siunaukseksi – tai ainakin mahdollisuudeksi.

Asiantuntijat ja median ammattilaiset odottavat tableteilta ja niille tehdyiltä sähköisiltä lehdiltä paljon. Media-ala on jälleen tilanteessa, jossa teknologian mahdollistama uusi julkaisuympäristö muokkaa julkaisemista ja koko mediakenttää [Harju *et al.,* 2011]. Tablet-laitteiden markkinat ovat viemässä mediatuottajia kohti uutta julkaisutapaa, jossa moni vanhaan printtiin – ja myös verkkoon – liittyvä asia pitää miettiä uudestaan. Toisaalta on panostettava digitaaliseen julkaisuun, mutta toisaalta on haastavaa saada lukijat tarttumaan uuteen tuotteeseen. Olemassa olevan lehtibrändin onnistunut julkaiseminen tabletilla vaatii kuitenkin teknologian tuomien mahdollisuuksien hyödyntämistä [Kawohl, 2011]. Median digitalisoituminen tarjoaa mahdollisuuksia luoda täysin uudenlaisia interaktiivisia, visuaalisesti näyttäviä ja helppokäyttöisiä tuotteita, joista saadaan myös tuloja.

Tablettien ja sähköisten lehtien yhteinen taival on kuitenkin parhaimmillaankin vasta alle kahden vuoden ikäistä. Tavallaan kaikki julkaisut elävät vielä pilottivaihetta [Harju *et al.,* 2011]. Siksi onkin kohtuutonta odottaa, että tabletit muuttaisivat journalismin perusasioita ja esitystapoja silmänräpäyksessä. Tilanne on vaikea, mutta yhdessä Internetin ja tablettien välityksellä journalismilla on myös voitettavaa – sillä on mahdollisuus uudistua parempaan suuntaan tekemällä itsestään helpommin saavutettavaa ja taloudellista. Samalla tulee kuitenkin muistaa, että jos sisällöstä halutaan veloittaa Internetin rajattoman ja ilmaisen informaation aikakaudella tarvitaan vastapainoksi laadukasta sisältöä. [Vehkoo, 2011]

Ilmassa on paljon epätietoisuutta, sillä päämäärät ovat vielä osittain hämärän peitossa. Vasta tablettien käyttäjämäärän kasvaessa alkanee niin mediataloille kuin kuluttajille itselleenkin hahmottua, minkälaista ja minkä näköistä sisältöä laitteille halutaan. Tulevaisuus näyttää, miten onnistutaan tekemään journalismia, joka parhaiten hyödyntää tablettien ominaisuuksia – teknisiä mahdollisuuksia, käytettävyyttä, visuaalisia erityispiirteitä, käyttäjien rooleja, käyttökonteksteja ja tablettien sijoittumista kuluttajien arkipäivään. [Harju *et al.,* 2011].

Juna kohti uudenlaista julkaisemista ja lukukokemusta on jo liikkeellä. Siinä junassa kannattaa lehtitalojen pysyä kyydissä, sillä jokaisella pysäkillä odottaa uusia matkustajia, jotka janoavat uusia lukuelämyksiä.

Viiteluettelo

[Ahlroth, 2011] Jussi Ahlroth, iPad research (2011). Available in: http://ipadresearch.wordpress.com/

[Anygraaf, 2011] http://www.anygraaf.fi/fin/etusivu/index.html

[Apple Press Info, 2010] Apple Press Info, Apple sells three million iPads in 80 days (22.6.2010). Available in: http://www.apple.com/pr/library.

[Bellis, 2008] Mary Bellis, Who invented touch screen technology? Available in: http://inventors.about.com/od/tstartinventions/a/Touch-Screen.htm

[Bershewsky *et al.*, 2011] Tapio Bershewsky, Tero Mäkikangas, Rami Lappalainen ja Mikko Hannula, Tietokoneen tulevaisuus on täällä. Tietokone (6/2011), 30-39.

[Black, 2010] Kevin Black, Magazines and the tablet interface. California Polytechnic State University, Faculty of Graphic Communication Department, Senior Project, 2010.

[Budiu and Nielsen, 20101] Raluca Budiu and Jacob Nielsen, Usability of iPad and websites, 2nd edition. Nielsen Norman Group Report, 2011. Available in: http://www.nngroup.com.

[Buitelaar, 2010] Michiel Buitelaar, iPad ei ole näköislehti. *Tietokone* (5/2010), 29.

[Castells, 1996], Manuel Castells, *The Rise of the Network Society*. Wiley-Blackwell, 1996.

[Dodge, 2010] Steve Dodge, Digital magazines and the iPad. *The Journal of Education, Community and Values* **10** (7/2010). Available in: http://bcis.pacificu.edu.

[eMedia, 2007] Taloustutkimuksen eMedia 2007-tutkimus.

[eMedia, 2011] Taloustutkimuksen aMedia 2011-tutkimus.

[Evans, 2011] Dean Evans, 10 memorable milestones in tablet history, In depth: 21 years before iPad, the GRiDPad was turning heads (31.1.2011). Available in: http://www.techradar.com.

[Garside, 2011] Juliette Garside, Nokia's Windows tablet to take on Apple's iPad. *The Guardian* (16.11.2011). Available in: http://www.guardian.co.uk.

[Hamilo, 2010] Marko Hamilo, Miten kosketusnäyttö toimii? *Tiede* (11/2010).

[Harju *et al*., 2011] Auli Harju, Anssi Männistö, Ari Heinonen, Debattia tableteista. Lukulaitejournalismi – nyt –tutkimushankkeen väliraportti. Journalismin, viestinnän ja median tutkimuskeskus, Tampereen yliopisto, 2011.

[Heikkilä, 2011] Harri Heikkilä, Onko iPad Gutenvergin hauta vai uudelleensyntymä? Tabletit julkaisujen ja suunnittelijoiden kannalta. Journalismin päivä (14.10.2011). Helsingin messukeskus.

[Helsingin Sanomat, 2011] Helsingin Sanomat, HS:n iPad-version ladannut jo 35 000 käyttäjää (15.10.2011), B11.

[Hermida, 2011] Alfred Hermida, The value of theory in driving innovation in journalism. Available in: http://www.reportr.net.

[Honkonen, 2011] Juha Honkonen, haastattelu 21.11.2011.

[Hyyrysalo, 2009] Sampsa Hyyrysalo, *Käyttäjä tuotekehityksessä. Tieto, tutkimus ja menetelmät*. Taideteollinen korkeakoulu, 2009.

[IHS, 2011] IHS, Global media tablet shipment forecast (8/2011). Available in: http://www.isuppli.com.

[ISO 9241-11, 1998] Ergonomic requirements for office work with visual display terminal. Part 11: Guidance of usability. Geneva: International Organisation for Standardization (1998).

[Itkonen, 2011] Markus Itkonen, Tabletin käyttöä on jo tutkittu. *Julkaisija* (3/2011), 8.

[Jobs, 2007] Steve Jobs, iPhone presentation. Macworld Conference & Expo, 2007.

[Jokinen, 2011] Pertti Jokinen, Hesarin uusi päätoimittaja: Rakasta iPadiani. *Julkaisija* (3/2011), 22-24.

[Jäppinen, 2011] Kaija Jäppinen, Tyyliä ja lukijan omia valintoja. *Suomen Lehdistö* **81** (1/2011), 16.

[Järvinen, 2010] Petteri Järvinen, Apple mullistaa sähköiset lehdet. *Tietokone* **15** (8/2010), 20.

[Kansallinen mediatutkimus, 2011] Kansallinen mediatutkimus, Suomalaiset lukevat lehtiään monella tavalla (2011). Saatavilla http://www.levikintarkastus.fi.

[Kauppalehti, 2011] Kauppalehti, Tablet-tietokoneilla kova vauhti: Myynti +280 prosenttia (20.10.2011). Saatavilla: http://www.kauppalehti.fi.

[Kawohl, 2011] Konstantin Kawohl, Magazines in a digital lunchbox: How can magazines on tablet devices like the apple iPad help evolve editorial design and reader experience? Berlin Technical University of Art, Intermediate Examination Paper, 2011.

[Kay, 1972] Alan C. Kay, A personal computer for children of all ages. In: Proc. of *ACM National Conference,* Boston (1972).

[Kettunen, 2011] Katriina Kettunen, Lehtitalot luottavat sähköisten lehtien imuun (21.3.2011). Saatavilla: http://yle.fi.

[Kinturi, 2011a] Marja-Liisa Kinturi, Sisältö lopulta ratkaisee. *Julkaisija* (3/2011), 27.

[Kinturi, 2011b] Marja-Liisa Kinturi, Kirjapaino etsii uutta. *Julkaisija* (3/2011), 35-38.

[Kivioja, 2011] Pasi Kivioja, Pyyhkimään. *Suomen Lehdistö* **81** (4/2011), 17.

[Lee and Tsai, 2011] Aaron Lee and Joseph Tsai, PC hardware players to phase out from tablet PC market in 2012. *Digitimes* (17.11.2011). Available in: http://www.digitimes.com.

[Malin, 2011] Veera Malin, Hesarin iPad-lehdestä versio 1.1. *Suomen Lehdistö* **81** (1/2011), 16.

[McCracken, 2010] Harry McCracken, The long fail: a brief history of unsuccessful tablet computers. *PC World* (1.2.2010). Available in: http://www.pcworld.com.

[Meyer, 2009] Philip Meyer, *The Vanishing Newspaper: Saving Journalism in the Information Age*. University of Missouri Press, 2009.

[Moss, 2011] Steve Moss, A bright future for eReader technologies. *MRS Bulletin* **36** (2011), 567-568. Available in: http://journals.cambridge.org.

[Nielsen, 1993] Jacob Nielsen, *Usability Engineering*. Academic Press, Boston, 1993.

[Nielsen Company, 2011] Nielsen Company, Connected devices: How we use tablets in the U.S. (5.5.2011). Available in: http://blog.nielsen.com.

[Pietilä, 2007] Antti-Pekka Pietilä, *Uutisista Viihdettä, Viihteestä Uutisia*. Art house 2007.

[Preece, 1995] Jenny Preece, *Human Computer Interaction*. Addison-Wesley, 1995.

[TNS Gallup, 2010] TNS Gallup, Sähköisten lukulaitteiden markkina, 2010. Saatavilla: http://www.viestinnankeskusliitto.fi.

[Ruulio, 2011a] Tiina Ruulio, Pääkirjoitus. *Julkaisija* **19** (3/2011), 5.

[Ruulio, 2011b] Tiina Ruulio, Uusi aika vaatii Uuden toimitusjärjestelmän. *Julkaisija* **19** (3/2011), 30-32.

[Sampola, 2008] Päivi Sampola, Käyttäjäkeskeisen käytettävyyden arviointimenetelmän kehittäminen verkko-opetusympäristöihin sopivaksi. Vaasan yliopisto, Teknillinen tiedekunta, tietotekniikan laitos. Monografia, Acta Wasaensia, 192.

[Steele, 2011] Chandra Steele, History of the tablet. *PC Magazine* (7.8.2011) . Available in: http://www.pcmag.com.

[Tuuli, 2011] Heikki Tuuli, KMT: lukulaitteet vielä vähissä. *Markkinointi & Mainonta* (15/2011).

[Vehkoo, 2011] Johanna Vehkoo, *Painokoneet seis!*. Teos, 2011.

[Vinter, 2011] Hannah Vinter, Mario Garcia: Newspapers need to carve their niche on tablets (2011). Available in: http://www.wan-ifra.org.

[Virranta, 2011a] Riikka Virranta, Rahtaus vaihtui lataamiseen. *Suomen Lehdistö* **81** (5/2011), 12-13.

[Virranta, 2011b] Riikka Virranta, Kansa vaatii näköislehteä tabletille. *Suomen Lehdistö* **81** (8-9/2011), 9.

[Virranta, 2011c] Riikka Virranta, iPad opetti tekemään verkkoa. *Suomen Lehdistö* **81** (8-9/2011), 13.

Pasianssia palaverissa – tehokas etätyö ohjelmistoalalla

Timo Virtanen

Tiivistelmä

Tietotyö on luovaa ja vahvasti ajatteluun perustuvaa. Tällainen aivoja kuormittava työ on hyvin löyhästi aikaan ja paikkaan sidottua. Siksi se on hyvin soveltuvaa toimiston ulkopuolella tapahtuvalle etätyölle. Ohjelmistoala on tietotyöpainotteista ja jo valmiiksi paljolti sähköistä. Tästä syystä askel etätyönä tehtävään ohjelmistotyöhön on lyhyt. Tämä tutkielma kokoaa yhteen tutkimuksia ohjelmistoetätyön tehokkuustekijöistä.

Avainsanat ja -sanonnat: Ohjelmistotuotanto, etätyö, tehokkuustekijät
CR-luokat: D.2.9

1. Johdanto

Tekniikan kehittyminen on lisääntyvissä määrin mahdollistanut työn tekemisen ajasta ja paikasta riippumatta. Tämä perinteisestä määrätyin kellonajoin työpaikalla tehtävästä työstä poikkeava työmuoto tunnetaan termillä etätyö. Etätyö voi olla osittaista, esimerkiksi päivä viikossa, tai täysipainoista, jolloin henkilö voi suorittaa koko työpanoksensa haluamastaan sijainnista käsin.

Etätyön perimmäinen tavoite on työnteon tehostaminen. Tehostuksen lähteenä on vapaus, jonka tarkoitus on mahdollistaa yksilökohtaisen tuottavimman ajankohdan käyttäminen työntekoon. Yritystasolla tällä etuudella pyritään metsästämään työntekijöiksi taitavia yksilöitä asuinpaikasta huolimatta. Luonteeltaan tämä on varsin poikkeava työn tekemisen malli ja antamistaan hyödyistä kumpuaa myös joukko haasteita. Ollakseen tehokasta etätyön järjestämiseen tulee kiinnittää erityistä huomiota sekä työtä tekevän että tarjoavan tahon puolelta niin kommunikoinnin kuin erilaisten taitojenkin muodossa [Larsen and McInerney, 2002].

Ohjelmistoalalla tehtävä ensisijaisesti ajattelua vaativa työ ei ole automaattisesti paikkaan sidottua. Esimerkiksi algoritmioptimointia voi hahmotella paperille junan ravintolavaunussa ja epäkäytettävän käyttöliittymän parannusidea saattaa iskeä lenkkeillessä. Koska tietokone on pääasiallinen työväline ja nopeat tietoverkkoyhteydet ulottuvat laajalle, etätyön tekeminen on alalla hyvin vaivatonta: saamansa idean voi yrittää toteuttaa välittömästi. Tämä on omiaan hämärtämään työ- ja vapaa-ajan eroa.

Tässä tutkimuksessa selvitetään, millaisia vaikeuksia ohjelmistoalalla suoritettava etätyö voi kohdata ja miten niitä voi ratkaista. Jotkin esitellyt aiheet ovat yleispäteviä kaikessa etätyössä, mutta tarkastelu tehdään ohjelmistoalan tarpeita ja mahdollisuuksia silmälläpitäen. Painopiste on ohjelmistoetätyön tehokkuuden tavoittelussa.

2. Etätyö ohjelmistoalalla

Nykyaikainen ohjelmistotyö on usein projektiluontoista. Projekti on työsuoritus, jolla on selkeä tavoite, alkamis- ja päättymisajankohdat ja jonka toteuttaminen määrätään usein yhden tai useamman tiimin harteille. Tiimi on elinkaareltaan lyhyehkö, tyypillisesti kuukausista vuosiin olemassa oleva, yksilöistä koottu työyksikkö. Tiimin jäsenet ovat vahvasti toistensa työpanoksista riippuvaisia. Heidät roolitetaan projektin tarpeen mukaisesti eri tehtäviin ja yksi tai useampi heistä osoitetaan projektipäälliköksi vastaamaan projektin läpiviennistä ja toimimaan yhteyshenkilönä tiimin ja muiden osakkaiden välillä. Osakkaat käsittävät tyypillisesti ainakin projektin tilanneen tahon, asiakkaan.

Yhdenkin tiimiläisen osa- tai kokoaikainen etätyöskentely muuttaa projektin luonnetta merkittävästi. Mikäli tiimissä on vähintään kaksi pääasiallisesti toisistaan ajallisesti tai sijainniltaan erillään työskentelevää henkilöä, jotka yhdessä työskentelevät yhteisen tavoitteen eteen pääsääntöisesti sähköisten viestimien yhdistämänä, on kyseessä etätyötiimi [Guido *et al.*, 2005]. Etätyötä tapahtuu tämän määritelmän mukaan jo silloin, kun työntekijät kahdessa eri toimipisteessä työskentelevät saman projektin parissa, vaikka työajat olisivat yhtenevät. Tämän tutkielman sisältö on sovellettavissa tällaisista tilanteista aina maailmanlaajuisesti eri aikavyöhykkeillä tapahtuvaan kehitystyöhön, sillä yhdistävä perusongelma syntyy kasvokkain kohtaamisen hankaluudesta.

Kuten perinteisten projektien kohdalla, niin etätyötiiminkin projektin onnistuneelle läpiviennille keskeistä on riskien tiedostaminen. Projektin epäonnistuminen myöhästymisen, budjettiylityksen tai vaatimuksia vastaamattoman lopputuloksen muodossa johtuu tyypillisesti riskienhallinnan pettämisestä. Tästä syystä etätyölle ominaisten riskien kartoitus ja tehokkuustekijöiden selvitys edesauttavat projektin onnistumista. Arvioituna yhdessä tunnettujen yksilö- ja yritystason hyötyjen ja haittojen kanssa ne antavat päätöksenteon perusteeksi riittävän tiedon siitä, kannattaako etätyö mahdollistaa.

2.1. Etätyön haasteet

Yksilötasolla vapaus ajasta ja paikasta tuo muassaan vaatimukset itsekurista ja sopeutumisesta sähköisesti viestimiseen. Toimistotuolin vaihtuminen laituriksi

ja työtovereiden muuttuminen pelkiksi nimimerkeiksi keskustelusovelluksessa ovat näkyvin ensivaiheen muutos. Harventuva tai olematon työpaikalla käynti voi aiheuttaa eristymisen tunteen ja vähentää sosiaalista kanssakäymistä [Guido *et al.*, 2005]. Satunnaiset kahvipöydässä käydyt keskustelut eivät ole lainkaan merkityksettömiä tiimihengen rakennuspalikoita. Kasvokkain käytävän keskustelun vaihtuminen sähköiseen, oli se sitten tekstimuodossa, puhelimitse tai videoneuvotteluin tapahtuvaa, aiheuttaa omat ongelmansa. Sähköisesti tapahtuva kommunikointi lisää väärinkäsitysten lisääntymisen ja konfliktien pahentumisen mahdollisuutta [Guido *et al.*, 2005]. Etänä töitä tekevälle on myös helppo sysätä useita vastuita eri tiimeissä ja projekteissa [Derosa and Lepsinger, 2010]. Tällöin ongelmaksi voi muodostua tiimiläisten roolien hämärtyminen ja aktiivisten projektien tavoitteiden keskinäisen priorisoinnin vaikeus [Guido *et al.*, 2005].

Yritykselle, joka mahdollistaa työntekijöidensä etätyön, syntyy teknisiä ja taloudellisia haasteita. Teknisiä vaikeuksia ovat tietoturvan ylläpito, kun töitä tulisi voida tehdä sijainnista riippumatta, sekä tehokkaaseen yhteydenpitoon ja projektiseurantaan tarvittavan teknologian järjestäminen [Guido *et al.*, 2005]. Taloudelliset ongelmat ovat seurausta ylimääräisistä etätyökeskeisistä koulutuskuluista sekä edellä mainitun teknologian kustannuksista [Guido *et al.*, 2005]. Mikäli yrityksessä seurataan tiettyä ohjelmistokehitysmallia, kuten ketterää kehitystä, voi tästä seurata erityisiä hankaluuksia. Esimerkiksi ketterä kehitys edellyttää säännöllistä yhteydenpitoa ja kannustaa sellaisiin toimintamalleihin kuin pariohjelmointi, jossa kaksi kehittäjää työskentelee saman tietokoneen äärellä. Etätyö ei kuitenkaan välttämättä estä tällaisen menetelmän noudattamista, kommunikointi siinä missä muutkin toimenpiteet kun onnistuvat tarvittaessa sähköisestikin. Esimerkiksi etätyötä tekeville soveltuvaan pariohjelmointiin on kehitetty apuohjelmisto [Schümmer and Lukosch, 2009].

2.2. Etätyön mahdollisuudet

Tunnetuin hyöty etätyöstä sitä tekevälle on itsenäisyys: joustavuus ajan ja paikan suhteen ja mahdollisuus hallita omaa ajankäyttöä [Guido *et al.*, 2005]. Myös työtyytyväisyys paranee, kun tiimiläinen voi vapaammin päättää miten kehittää ja organisoida työtään [Acuña *et al.*, 2008].

Yrityksen etätyöstä saamat strategiaedut ovat kiistämättömät. Etätyöläinen ei ole sidottu sijaintiin, joten työntekijän valinnassa voidaan painottaa ammattitaitoa muuttohalukkuuden ylitse ja saada samalla avattua läheisempi kontakti alueellisiin toimijoihin [Guido *et al.*, 2005]. Esimerkiksi eri puolilla maata asuvat työntekijät voivat kukin ylläpitää lähikaupunkinsa kattavaa asiakasverkostoa.

Paikallisella tasolla minimoituvat näin matka- ja toimistokulut [Guido *et al.*, 2005]. Globaalissa mittakaavassa yritys voi etätyöntekijöitä hankkimalla saavuttaa ympärivuorokautisen työskentelyn projektin parissa ja henkilöstönsä tavoitettavuuden tekemättä suuria kiinteitä investointeja [Guido *et al.*, 2005]. Ammattitaidon painotus valintaperusteena saa lisäpontta väitteestä, jonka mukaan etätiimin kokoaminen tapahtuu kasvokkain työskentelevää tiimiä todennäköisemmin tehtävän vaatimien taitojen kuin etnisyyden, fyysisen viehättävyyden tai asenteiden yhteensopivuuden perusteella [D'Souza and Colarelli, 2010].

3. Etätyön tehokkuustekijät

Tuoreen tutkimuksen mukaan neljännes etätyötiimeistä alisuorittaa syistä, jotka olisivat ratkaistavissa [Derosa and Lepsinger, 2010]. Tämä luku käsittelee projektin vaiheita viitekehyksenä käyttäen syitä tiimin tehottomuudelle ja tarjoaa malleja näiden korjaamiseksi. Tässä jaottelussa ensimmäisenä tulee projektia edeltävä esivaihe, jolloin määritellään ne yrityksen etätyötä koskevat linjaukset, joita myöhemmissä projekteissa tullaan käyttämään. Toisena seuraa itse projektin alkuvaihe, jolloin projektin luonne selvitetään ja työstävä tiimi kootaan. Kolmantena on keskivaihe, kun projektin työstö on käynnissä ja tiimi on altis vastoinkäymisille. Neljäs ja viimeinen vaihe on loppuvaihe, jolloin projekti valmistuu ja tiimin tekemä työ arvioidaan.

3.1. Projektin esivaihe

Etätyön tehokkuus sanellaan jo projektin esivaiheessa. Jotta etätyöllä tehtävää projektia kannattaa harkita, on yrityksellä syytä olla olemassa oleva suunnitelma etätyön tukemisesta. On tärkeää ymmärtää etätyön luonteesta johtuva painopisteiden erilaisuus, kuten kommunikoinnin korostuva tärkeys. Mikäli yritys yrittää vain siirtää perinteisesti käytetyn työskentelymallin etätyöpohjalle, luvassa on suorituskykyongelmia [Derosa and Lepsinger, 2010]. Yrityksen yleisesti noudattamat prosessit, kuten säännöllisten tilanneraporttien vaatiminen, voivat sellaisenaan etätiimeille sovellettuina rampauttaa tehokkuutta [Piccoli and Ives, 2003]. Kaikkien projektin osakkaiden on ymmärrettävä etätyönä tehtävän projektin tavoitteet, jotta se voi onnistua [Pokharel, 2011].

Yrityksen ylenmääräisillä teknologialinjauksilla on helppo vesittää etätyön tehokkuus. Vaikka ennen kaikkea kommunikointityökalut ovat tarpeen etätiimin toiminnan mahdollistamiseksi, teknologia yksin ei tee etätyöstä tehokasta [Lurey and Raisinghani, 2001]. Ei ole tavatonta, että suurella rahalla etätyöhön tehdyt panostukset hukkuvat tehottoman projektipäällikön, vastuuntunnotto-

mien tiimiläisten, tiimiin panostamiseen tarvittavan ajan puuttumisen tai heikon koulutuksen johdosta [Derosa and Lepsinger, 2010]. Jotta sijoitukset kantaisivat hedelmää, kannattaa apuvälineissä painottaa sellaisia työkaluja, joita tiimiläiset käyttävät entuudestaan ja jotka ovat upotettavissa jokapäiväiseen työntekoon: esimerkiksi sähköposti, internetselain ja webbikamera [Nunamaker *et al.*, 2009].

Mikäli yhteisössä on palkitsemisen kulttuuri, on se syytä räätälöidä myös etätyöympäristössä noudatettavaksi. Etätyötiimeissä tehokkaaksi on havaittu tiimiperusteinen palkitsemismalli [Lurey and Raisinghani, 2001]. Malli, jossa palkkiosta päättävät esimiesten lisäksi myös vertaiset, toimii tehokkaasti ryhmätöissä laiskottelun torjumisessa [Bryant *et al.*, 2010]. Tällainen malli voi perustua säännöllisesti tehtävän tehokkuuskyselyn ja -arvioinnin tuloksiin, mutta ohjelmistoalalla pohjatietoa saa jo seuraamalla tiimiläisten osallistumista projektin osa-alueisiin.

3.2. Projektin alkuvaihe

Projektia työstämään valittavan tiimin koostumus on suuri suorituskykytekijä. Tiiminvalinnassa on huomioitava useita seikkoja tiimin koosta yksilöiden luonteenpiirteisiin asti. Etätiimin kokoamisessa on perusteltua käyttää samoja painopisteitä kuin tavallistenkin tiimien muodostuksessa [Lurey and Raisinghani, 2001]. On olemassa kompastuskiviä, joihin ei kannata tiimiä kootessa sortua. Vaikka etätyö joustavine läsnäolovaatimuksineen mahdollistaa yksilön sitomisen useisiin tiimeihin samanaikaisesti, ei tätä kannata tehdä ilman painavia perusteita. Tehokas ajankäyttö kärsii, jos yksilö joutuu painottamaan useiden eri tehtävien ja velvollisuuksien välillä. Lisäksi tiimikoko on syytä pitää pienenä. Nyrkkisääntönä useampi viiden hengen tiimi on parempi ratkaisu kuin yksi monikymmenpäinen. Jos yksi pieni tiimi ei riitä, voi olla viisasta määrittää erillinen ydintiimi ja tälle neuvoa-antava aputiimi [Derosa and Lepsinger, 2010]. Yhteyshenkilöt määrittämällä minimoidaan myös tarvittavan kommunikoinnin määrää.

Tiimiin ei tule määrittää suhdeperustein henkilöitä, jotka eivät aio työskennellä tavoitteiden vuoksi: jos tiimiläiset eivät osaa nimetä kaikkia muita tiimiin kuuluvia, on tunnistamattomien joukossa luultavasti yksi tai useampi tällainen henkilö [Derosa and Lepsinger, 2010]. Kun tiimiä kootaan, kannattaa se rakentaa pysyvyys mielessä, sillä tiimiläisten vaihtuvuus johtaa helposti huonoon suorituskykyyn [Derosa and Lepsinger, 2010]. Henkilövalinnassa painopisteen on oltava henkilön osaamisella projektin tarpeet huomioiden [Lurey and Raisinghani, 2001]. Kuitenkaan tiimiin ei tule valita yksin teknisen osaamisen

vuoksi: etätyöhön soveltuviksi luonteenpiirteiksi on havaittu aloitteellisuus, joustavuus, hyvä kommunikointikyky ja edistyneet ihmissuhdetaidot [Derosa and Lepsinger, 2010].

Työntekijän persoonallisuudellakin on merkitystä. Persoonasta riippuu, kuinka yksilö käsittää häneen vaikuttavien tapahtumien hallintamahdollisuuden. Hallintakäsitys jakaa ihmiset kahteen ryhmään: sisäisen ja ulkoisen hallintakäsityksen omaaviin. Sisäiset ovat itseohjautuvia ja uskovat kykenevänsä ohjaamaan elämäänsä, kun ulkoiset vuorostaan uskovat heitä koskettavien tapahtumaketjujen olevan muiden käsissä. Se, mihin kohtaan tiimiläinen tällä akselilla asettuu, vaikuttaa siihen, tulkitseeko hän ylhäältä ja sivulta tulevat päätökset ja toimet mahdollistavina vai rajoittavina. Tämä heijastuu siihen, kuinka yksilö käyttäytyy muita kohtaan. Esimerkiksi sisäisen hallintakäsityksen omaava suhtautui innokkaasti mahdollisuuteen tehdä etätyötä, kun taas ulkoisen hallintakäsityksen edustaja lähestyi asiaa negatiivisen kautta. Toisaalta, vaikka sisäisen hallintakäsityksen edustajat ovat innokkaan asennoitumisen ja itsevarman omiin kykyihinsä uskomisen johdosta teoriassa täydellisiä henkilöitä itseohjautumista vaativan etätyön tekemiseen, on tälläkin ääripäällä riskinsä. Omien kykyjen yliarviointi voi johtaa aikaa vieviin parannushankkeisiin, jotka eivät milloinkaan valmistu. Esimerkkinä tällaisesta on vanhasta ohjelmakoodista eroon pyrkiminen tekemällä tilalle uutta, parempaa koodia, joka ei kuitenkaan lopulta vastaa laadultaan alkuperäistä. Ratkaisuna hallintakäsityksen haitallisten vaikutuksien minimointiin toimii sisäisen tapauksessa säännöllinen projektin tilan ja tiimin tehokkuuden arviointi ja ulkoisen tapauksessa etätyön tekoon perehdyttävän koulutuksen järjestäminen, sekä etätyöstä kokemusta omaavan henkilön liittäminen tiimiin. [Lee-Kelley, 2006]

Tiimin kokoamisen jälkeen jäsenten roolit tulee määritellä selkeästi [Lurey and Raisinghani, 2001]. Roolien määrityksessä kannattaa käyttää tarkkuutta summittaisuuden sijasta. Hierarkkisten rooliensuhteiden tilanteessa on huomioitava etätyön tapauksessa tavoitettavuus, tarvittaessa määrittämällä helpommin tavoitettava kontaktihenkilö aiemman tilalle. Tehokkuus kärsii, mikäli henkilö ei tiedä, kehen olla yhteydessä, tai mikäli tavoiteltava henkilö sijaitsee eri aikavyöhykkeellä ja yhteydenpitoon aiheutuu tästä ylimääräistä viivästystä [Derosa and Lepsinger, 2010].

Rooleista korostuvin on projektipäällikön rooli. Hänen tehtävänään on huolehtia projektin tehokkaasta läpiviennistä. Etätiimin projektipäällikön asema ei ole helppo, sillä johtaminen on yksi keskeisimmistä suorituskykyyn vaikuttavista tekijöistä. Tiimin johtajan valinnassa tulee käyttää harkintaa, sillä etäjoh-

taminen edellyttää erityisesti ihmissuhdetaitoja ja taitoa tasapainoilla niiden ja tehokkaan toiminnan välillä [Derosa and Lepsinger, 2010]. Taito motivoida on ihmissuhdetaidoista keskeinen, mutta myös kyky eettiseen johtamiseen on hyödyksi [Lee, 2008]. Ei olekaan ihme, että transformaalinen johtaminen, eli ihmiskeskeinen johtamistyyli, toimii etäympäristössä perinteistä ympäristöä tehokkaammin [Purvanova and Bono, 2009]. Ihmiskeskeinen johtamistyyli painottaa yksilön huomioonottamista, älyllistä stimulointia, inspiroiden motivointia ja ihannoiden vaikuttamista. Oli johdettava tiimi sitten etäinen tai perinteinen, ihmiskeskeisen johtamistyylin havainnointi johtajassa lisää tiimin jäsenten tyytyväisyyttä projektiin [Purvanova and Bono, 2009]. Yllättävästi viestintään käytettävä teknologia ei vaikuta tämän johtamistyylin tehokkuuteen: ei ole merkitystä, perustuuko kommunikointi tekstipohjaiseen viestinvaihtoon vai videoneuvotteluihin [Hambley *et al.*, 2007].

Projektipäällikön tehtävänä on määrittää, jakaa, osoittaa ja aikatauluttaa projektin osa-alueet [Pokharel, 2001]. Tämä työ on tehtävä yhtä perusteellisesti kuin perinteisen projektinkin tapauksessa, jotta projektiin sitoutuminen on mahdollista [Guido *et al.*, 2005]. Jotta kommunikointi on mahdollisimman tehokasta, kannattaa tiimin sopia keskenään projektin tiimoilta käytettävistä standardeista ja terminologiasta [Nunamaker *et al.*, 2009]. Alusta alkaen on projektin päätavoitteet tehtävä yksikäsitteisiksi [Lurey and Raisinghani, 2001]. Näiden pohjalta tulee määrittää työstettävän kokoiset tehtävät valmistumispäivämääräehtoineen tiimiläisille osoitettaviksi [Larsen and McInerney, 2002]. Projektin alussa kannattaa suunnitella tehtäviä jotka vaativat vuorovaikutusta, sillä niillä on yhteenkuuluvuutta tukeva vaikutus [Guido *et al.*, 2005]. Myöhemmin tämäntyyppisten tehtävien määrää voi vähentää kustannustehokkuuden niin vaatiessa.

Projektin voi käynnistää siten, että tiimi saa räjähtävän lähdön työntekoon. Koko etätyötiimi kannattaa järjestää paikalle kasvokkain tapahtuvaan käynnistystapahtumaan. Vaikka tästä koituukin kustannuksia, ovat tehokkuushyödyt kiistämättömiä: tiimit, joiden jäsenet tapaavat toisensa pian tiimin muodostumisen jälkeen, ovat tehokkaampia kuin tiimit, jotka eivät milloinkaan tapaa kasvokkain [Derosa and Lepsinger, 2010]. On havaittu, että joillakin ihmisillä on vaikeuksia luottaa henkilöihin, joita ei ole milloinkaan tavannut [Larsen and McInerney, 2002]. Kasvokkain tapaaminen helpottaa roolien lujittamista ja luottamuksen syntyä ja avaa mahdollisuuden muutoin harvemmin syntyvälle, projektin aiheista poikkeavalle, epäviralliselle keskustelulle. Etätyötiimien tuleekin olla ensisijaisesti tiimejä yhteisin tavoittein: henkilösuhteiden tulee olla vahvoja, jotta tavoitteiden saavutus on mahdollista [Lurey and Raisinghani,

2001]. Luottamuksen syntyminen on kriittistä tehokkaan tiimin syntymiselle [Larsen and McInerney, 2002]. Se myös ehkäisee pelon ja väärinkäsitysten kärjistymistä virtuaaliympäristössä [Lee, 2008].

Projektissa noudatettavien prosessien tulee olla vakiintuneita [Lurey and Raisinghami, 2001]. Tämä tarkoittaa sitä, että työvaiheisiin voi tukeutua ja ne edistävät tavoitteiden saavuttamista. Suositeltavaa on järjestää projektin keston ajaksi virtuaalinen hengailupaikka, jossa tiimiläiset voivat tavataan toisiaan epävirallisissa merkeissä. Tällaisessa palvelussa voidaan myös jakaa projektin edistymiseen liittyvää tietoa, joka on omiaan kannustamaan muitakin antamaan panoksensa [Nunamaker *et al.*, 2009]. Samaan yhteyteen voidaan myös koota resurssipankkia tiimiläisten kokemuksista eri tilanteissa ja kohtaamista ongelmista ratkaisuineen [Derosa and Lepsinger, 2010]. Järjestelmän sitä tukiessa anonyymin keskustelun salliminen madaltaa kynnystä avoimeen ja suoraan keskusteluun [Nunamaker *et al.*, 2009].

Käytettävää yhteydenottovälinettä kannattaa miettiä tilannekohtaisesti. Videoneuvottelua kannattaa välttää silloin, kun ilmoitus sähköpostitse riittäisi. Kuitenkin videoneuvottelu on tehokkaampi menetelmä ongelmanratkaisulle ryhmissä. Pikaviestimilläkin on sijansa kommunikointimenetelmissä, sillä ne avaavat mahdollisuuden spontaaniin keskusteluun. Erilaisia palveluita ja yhteydenpitovälineitä ei kuitenkaan kannata järjestää liikaa, sillä niiden runsas määrä on alisuorittavia tiimejä yhdistävä tekijä. [Derosa and Lepsinger, 2010]

Varhaisen vaiheen säännöllinen tehtäväkeskeinen kommunikointi on merkittävä tekijä transaktiivisen muistin luomisessa. Transaktiivinen muisti tarkoittaa sitä, että tiimin jäsenet erikoistuvat erilaisiin asioihin, tietävät mihin asioihin tiimitoverit ovat erikoistuneet ja tukeutuvat aihepiirin asiantuntijaan apua kaivatessaan. Tämä järjestely on omiaan luomaan tehokkaan, ammattitaitoisen ja tasa-arvoisen tiimin. Kun yksilöt oppivat luottamaan toisiinsa, seurauksena on tehokkuuden lisääntyminen. Sijoitus on kestävä: vaaditaan projektin elinkaaren keskivaiheella tapahtuva suorituskykyongelma ennen kuin usko vertaisten osaamiseen horjuu. Kun transaktiivinen muisti on muodostunut, tehtäväkeskeisen kommunikoinnin merkitys tehokkuuden edistäjänä heikkenee tehtäväkeskeisen tiedon koordinoinnin noustessa avaintekijäksi. Tämä helpottaa kommunikointia, sillä kehittyneen transaktiivisen muistin muodostanut tiimi selviää vähemmällä keskustelulla. Samoin toimii esimerkiksi hyvin yhteenpelaava jääkiekkoketju. Koska etätyö ei ole este transaktiivisen muistin syntymiselle, on sen muodostumiseen tähtääminen kannattavaa. [Kanawattanachai and Yoo, 2007]

3.3. Projektin keskivaihe

Projektipäällikön keskittyminen oikeisiin asioihin on onnistuvan projektin edellytys. Tärkeitä asioita ovat projektia koskevan tiedon välittäminen ja tiimin tehokkuuden tarkkailu ja kehittäminen. Suorituskyvyn säännöllinen tarkkailu paitsi omin mittarein, myös tiimiläisten ja projektin muiden osakkaiden antamien tietojen pohjalta auttaa havaitsemaan ongelmia ennen kuin ne pahenevat [Derosa and Lepsinger, 2010]. Etätyöläisillä on usein perinteisesti työskenteleviä vähemmän motivaattoreita, niin kuviteltuja kuin todellisiakin: virtuaalisessa työympäristössä jää helposti paitsi arvostuksesta, jota saa jäämällä tarpeen tullen ylitöihin ja saapumalla toimistolle etuajassa seuraavana päivänä [Nunamaker *et al.*, 2009]. Etätyössä tällainen panostaminen jää helposti näkymättömäksi, eikä yhteisvoimin läpi yön tehty uurastus saa osakseen samanlaista päätöstä kuin pimennetyn toimiston oven sulkeutuminen viimeisten poistuessa työpaikalta. Tämän tiedostaminen auttaa motivointia suunnitellessa.

Etätiimin tehokkuuden kannalta keskeistä on tukea kommunikointia. Kommunikointi itsessään ei ole etätyössä nouseva uudenlainen riski, vaan aivan vastaava kuin tavallisessa työmuodossakin: kommunikoinnin vähäisyys tai heikkolaatuisuus ovat avainriskejä työntekijöiden sijainneista riippumatta [Reed and Knight, 2010]. Riski on realisoitunut silloin, kun kommunikoinnin merkitystä vähättelee: tehokkaiksi havaittujen etätyötiimien jäsenet pitivät kommunikointitaitoa keskeisenä kehittämiskohteena, siinä missä alisuorittavat tiimit peräänkuuluttivat päätöksenteko- ja vastuunkantotaitojen olevan tärkeimpiä parannuskohteita [Derosa and Lepsinger, 2010]. Syynä tähän voi olla se, että tehottomammat tiimit eivät ymmärrä kommunikoinnin tärkeyttä ja huonosti kommunikoimalla kokevat vastoinkäymisiä, joita ei muutoin aiheutuisi [Derosa and Lepsinger, 2010].

Tiimihenki syntyy ihmisten keskinäisestä tuntemisesta ja luottamuksesta muiden taitoihin. Tiimistä puuttuva me-henki on merkki ongelmista. Tästä on kyse silloin, kun tiimin asioista puhutaan yksikössä, tiimiläiset eivät tunne toisiaan, ovat avoimen negatiivisia tai kyvyttömiä löytämään luovuutta tiimitoveriensa tekemisistä [Derosa and Lepsinger, 2010]. Perinteisissä työtiimeissä tutustuminen tapahtuu helposti lounas- ja kahvitauoilla [Lurey and Raisinghani, 2001]. Koska etätyöympäristössä tämä ei tapahdu yhtä luontevasti, on projektin edetessäkin syytä hyödyntää aktiviteetteja, joissa ihmiset tutustuvat toisiinsa. Kasvokkain keskustellessa esiintyvien visuaalisten apujen puuttuessa keskustelukumppanin viestiä on kuitenkin vaikeampi tulkita. Tämän johdosta määrätietoinen keskustelija tulkitaan herkästi vihaisena ja sovitteleva keskustelija iloisena, kun viestintä on tekstipohjaista [Cheshin *et al.*, 2011]. Tällaisten

virhetulkintojen syntymistä voi ehkäistä käyttämällä rikkaampia keskustelukanavia ja opettelemalla tuntemaan kanssakeskustelijat paremmin [Lurey and Raisinghani, 2001]. Tätä edesauttaa kun kasvokkain tapaamisen perinnettä jatketaan säännöllisesti vähintään vuosittain [Derosa and Lepsinger, 2010].

Etätyön konfliktit aiheutuvat usein väärinkäsityksestä tai vähäisestä kommunikoinnista, eivätkä niinkään aggressiivisesta käyttäytymisestä [Guido *et al.*, 2005]. Koska etätiimiläiset ovat harvemmin läsnä samanaikaisesti, on konfliktien havaitseminen vaikeaa. Tästä syystä etätyöympäristölle tyypillisien konfliktien välttämiselle ja hallinnalle on hyvä tarjota ohjeistusta [Guido *et al.*, 2005]. Pyrkimys konfliktien välttämiseen on perusteltua, sillä niiden suuren määrän on havaittu korreloivan huonon työviihtyvyyden kanssa, vaikka syy–seuraussuhde ei olekaan selvä [Acuña *et al.*, 2008]. Varmaa on kuitenkin se, että tyytyväiset etätyöläiset kuuluvat usein tehokkaaseen tiimiin [Guido *et al.*, 2005]. Työviihtyvyyteen satsaaminen on siten aina viisasta.

Kiireisessä ympäristössä projektin vaatimukset voivat muuttua hetkessä toisiksi yritysten muuttuneiden tilanteiden seurauksena. Tällöin on tärkeää, että uudet tavoitteet tuodaan selkeästi esille koko tiimille [Derosa and Lepsinger, 2010]. Ymmärrys vaatimusten vaihtumiseen johtaneista syistä ja selvyys niihin reagointiin tarvittavista muutoksista ehkäisee projektin ohjautumista sivuraiteille [Pokharel, 2011]. Muutosten ilmaisu selkeästi on tärkeää kautta linjan. Yksikäsitteisen ymmärtämisen tukemiseksi tulee varmistua siitä, että työprosessien ja konseptien kuvailuissa käytetään riittävää huolellisuutta [Nunamaker *et al.*, 2009]. Koko tiimin voimin sovitut käsitteet ovat tärkeitä. Sekaannusten aktiivinen välttäminen kannattaa, sillä virtuaaliympäristössä epäselvyyksien selvittäminen on hankalaa [Nunamaker *et al.*, 2009].

Tiedonkulun ongelmat ovat etätyössä perinteistä työmallia merkittävästi suurempi riski [Reed and Knight, 2010]. Tiedonkulku käsittää paitsi projektin tilan ja tavoitteiden välityksen, myös tiedon siitä, mitä työntekijältä itseltään odotetaan ja kuinka hän suoriutuu verrattuna muihin [Guido *et al.*, 2005]. Projektinhallintatyökalu, johon on koottu kaikki keskeinen projektin tieto ja joka on kaikkien osallisten saatavilla, on tehokas tapa välittää tiimiläisille kaikki heidän tarvitsemansa tieto. Työkalusta käy esimerkiksi verkkosivusto, jonne projektipäällikkö voi kirjata projektin keskeisiä piirteitä ja johon tiimin jäsenet saattavat luoda merkintöjä työsuorituksistaan.

Palaveritkin hyötyvät projektinhallintatyökalun tehokkaasta käytöstä. Kun palveluun yhdistetään tieto palaverien ajankohdista, käsiteltävistä asioista ja jälkikäteen muistiinpanoista, on myöhemminkin selvitettävissä, mitä milloinkin on päätetty. Etänä pidettävissä palavereissa keskeistä on hyvissä ajoin, vähin-

tään paria päivää ennen ajankohtaa, parhaan palaveriajankohdan selvitys ja palaverista tiedottaminen, sekä asialistan määrittäminen. Säännöllisesti vähintään kerran viikossa pidettävä palaveri yhdistää valtaosaa tehokkaista etätiimeistä. Palaverien rakenne on oleellinen tekijä. Tehokkaisiin tiimeihin muodostuu usein järjestely, jossa kukin tiimiläisistä on vuorollaan vetovastuussa palaverin läpiviennistä. Lisäksi, koska tiimiläiset voivat asua eri aikavyöhykkeillä tai heillä voi olla erilainen vuorokausirytmi, on kohteliasta kierrättää kokousaikaa tarvittaessa siten, etteivät samat henkilöt joudu toistuvasti muita useammin näkemään ylimääräistä vaivaa kokoukseen osallistuakseen. Palavereille on myös syytä määrittää maksimikesto, josta pidetään kiinni. Tällä estetään tilaisuuden venyminen rönsyilevän keskustelun myötä kaikkien osallisten ajanhukaksi. [Derosa and Lepsinger, 2010]

Huono johtaminen on helppo tunnistaa, mutta tilanteen muuttaminen on vaikeaa. Kun projekti ei täytä tavoitteitaan, tiimin ja projektipäällikön välit ovat huonot, projektipäällikkö ei ole varma tiimin tavoitteista tai kun projektipäällikkö kuluttaa enemmän voimavaroja tiiminsä paikallisten edustajien kuin muiden työntekijöiden kanssa, on kyseessä huono johtaminen [Derosa and Lepsinger, 2010]. Oireena voi olla myös ajankohtaisen tehtävän huomiotta jättäminen, tai jopa hankaluus muodostaa yhteistä ymmärrystä itse tehtävän luonteesta [Nunamaker *et al.*, 2009]. Huonoon johtamiseen ei ole yhtä ratkaisua, mutta ongelmaa voi lähteä purkamaan useilla keinoilla: projektipäällikkö voi edesauttaa tilannetta olemalla tavoitettavissa, esittämällä selkeitä ajantasaisia tavoitteita ja parantamalla yhteishenkeä yhteisöllisyyden muodossa, esimerkiksi suunnittelemalla strategia koko tiimin voimin [Derosa and Lepsinger, 2010]. Tilanteen toistumisen ennaltaehkäisy on suositeltavaa. Tätä voi edistää määrittämällä yksiselitteisesti, mitattavasti ja realistisesti koko tiimiä koskettavia tavoitteita sekä asettamalla tiimiläisiä vastuuta kantavaan asemaan [Derosa and Lepsinger, 2010]. Epäonnistumisriskin minimoimiseksi tulee annettujen tehtävien edistymistä kuitenkin myös seurata [Pokharel, 2011].

Palautteen kerääminen ja välittäminen auttavat tehokkuusongelmien korjaamisessa. Etätyötä koskeva palaute on tärkeää, koska yksilön etätyönä tekemää suoritusta voidaan muissa yhteyksissä väheksyä perinteisesti tehtyä työpanosta huonompana [Nunamaker *et al.*, 2009]. Tehokkuutta koskevaa palautetta kannattaa jakaa säännöllisesti, konkreettisesti ja oikea-aikaisesti koko tiimin kesken [Guido *et al.*, 2005]. Tämä yleisestikin pätevä ohje on erityisen tärkeä etätyöläisille, jotka saavat näin arvokasta tietoa vertaistensa suorituksista: julkisen palautteenannon kautta tietoon saatava työtovereiden tehokkuus edistää luottamusta ja vähentää motivaatiota heikentävän hyväksikäytön

tuntemuksia [Guido *et al.*, 2005]. Tiimiläinen, joka tietää muiden tekevän oman osuutensa, kokee lisääntyvää yhteenkuuluvuuden tunnetta. Jotta kynnys palautteen välittämiseen pysyisi pienenä, palautteenantohetki voidaan yhdistää esimerkiksi säännöllisen palaverin yhteyteen. Projektipäällikölle on myös syytä toimittaa palautetta johtamistavastaan, jotta hän voi edelleen kehittyä johtajana [Derosa and Lepsinger, 2010]. Mikäli niin työtovereilta kuin projektin osakkailtakin kerätään palautetta, jossa todetaan tiimin menestyneen hyvin, on tiimin palkitseminen paikallaan. Palkitsemiseen riittää vähimmillään sanallinen tunnustus hyvin tehdystä työstä [Derosa and Lepsinger, 2010]. Julkisesti ja oikein perustein annettuna palkitseminen voimistaa tiimin tehokkuutta.

Koulutuksen avulla voidaan kehittää niitä osa-alueita, joilla tiimin on todettu alisuorittavan. Etätiimille osoitetun koulutuksen aihe voi olla sama kuin perinteisenkin tiimin, kuten uuteen ohjelmointikieleen perehdyttäminen, mutta myös etätyökeskeiset aiheet ovat hyviä [Guido *et al.*, 2005]. Nimenomaan etätiimeille arvokas taito on itseohjautuvuus: jos tiimille on osoitettu tilapäinen ohjaaja tai asiantuntija, hänelle voi antaa tavoitteeksi kouluttaa tiimiä niin, että se pärjää ilman hänen apuaan [Nunamaker *et al.*, 2009]. Toiveaiheita voi kysyä tiimiltäkin. Jotta koulutus on pidemmällä aikavälillä kehittävää, tulee jälkikäteen selvittää tapahtuman hyödyllisyys. Varsinkin oletettuun tarpeeseen vastaava koulutus, joka on järjestetty etänä, voi olla laadultaan todella huonoa, esimerkiksi ainoastaan näyttöpäätteen tuijottamista edellyttävää [Derosa and Lepsinger, 2010]. Tästä on seurauksena tiimiläisten kouluttautumisinnon latistuminen.

3.4. Projektin loppuvaihe

Projektin päättyessä on luonnollista analysoida sen tuloksellisuus ja tiimin tehokkuus. Tiimit ovat luonteeltaan tilapäisiä, joten menestynytkin tiimi voidaan purkaa projektin päätyttyä. Tässä kannattaa kuitenkin käyttää harkintaa, sillä kolme vuotta yhdessä ollut etätiimi on keskimäärin tehokkaampi kuin vain vuoden vanha [Derosa and Lepsinger, 2010]. Tämä saattaa johtua yksinomaan siitä, että tehottomat tiimit on purettu tilanteen jatkuessa kestämättömänä, mutta tulosten puolesta pitkäkestoisuuteen kannattaa tähdätä.

Loppuunsaatettu projekti antaa aina aihetta juhlaan. Vuosia tehokkaasti toimineen etätiimin voi pitää yhtenevänä yhä uudelleen tämänkaltaisen saavutuksen juhlistaminen kasvokkain [Derosa and Lepsinger, 2010].

4. Yhteenveto

Ohjelmistoala ja etätyöt ovat tehokas yhdistelmä. Huolimatta etätyön mukanaan tuomista pulmista, on tällä tavoin mahdollista suorittaa töitä yhtä tehokkaasti kuin perinteisin paikallisin menetelminkin. Tämän saavuttamiseksi etätyö vaatii kuitenkin erilaista kohtelua. Huolimatta siitä, millaista etätyön voisi kuvitella olevan, tiimihenki ja kommunikointi ovat hyvin tärkeitä osia. Kasvotusten tapahtuvan työskentelyn vähäisyys pitää korvata keskustelemalla runsaammin ja selkeämmin sähköisesti.

Yrityksen on ennen etätyön tarjoamista syytä selvittää, onko siirtymä sille todella hyödyksi ja pystyykö se tukemaan työntekijöitään kyllin hyvin. Tärkeää on tunnistaa, että saadakseen etätyöstä kaiken hyödyn irti on siihen panostettava. Työmuotojen eroista johtuen etätyötä ei voida sovittaa paikallisesti tapahtuvan työn raameihin, vaan on mietittävä, millä tavoin etätiimi kootaan, kuinka sen yhteishenkeä voi pitää korkealla ja miten taataan toimiva projektinhallinta.

Tämä tutkielma loi katsauksen erilaisiin ongelmakohtiin ja hyväksi havaittuihin ratkaisuihin, joilla tehokkuutta voisi lisätä. Asetelma tällaisena on suurelta osin etätyötä tarjoavan yrityksen etua tavoitteleva. Etätyötä tekevien työntekijöiden paikallisia tehokkuustekijöitä ei ole kuvattuna. Tarkastelematta jäi esimerkiksi se, kannattaako kokoaikaisen ohjelmistoetätyöläisen räätälöidä itselleen kotitoimisto ja kuinka työaika kannattaa sovittaa vapaa-ajan keskelle. Käsiteltynä on näin ollen vain toinen puoli etätyöstä. Tällaisesta etätyön henkilökohtaisesta puolesta toivoisin näkeväni vastaavanlaisen katsauksen.

Viiteluettelo

[Acuña *et al*., 2008] Silvia T. Acuña, Marta Gómez and Natalia Juristo, How do personality, team processes and task characteristics relate to job satisfaction and software quality? *Information and Software Technology* **51,** 3 (March 2009), 627–639.

[Bryant *et al*., 2010] Stephanie M. Bryant, Susan M. Albring and Uday Murthy, The effects of reward structure, media richness and gender on virtual teams. *International Journal of Accounting Information Systems* **10**, 4 (December 2010), 190–213.

[Cheshin *et al*., 2011] Arik Cheshin, Anat Rafaeli and Nathan Bos, Anger and happiness in virtual teams: Emotional influences of text and behavior on others' affect in the absence of non-verbal cues. *Organizational Behavior and Human Decision Processes* **116**, 1 (September 2011), 2–16.

[Derosa and Lepsinger, 2010] Darleen M. Derosa and Richard Lepsinger, *Virtual Team Success: a Practical Guide for Working and Leading from a Distance*. John Wiley and Sons, 2010.

[D'Souza and Colarelli, 2010] Geeta C. D'Souza and Stephen M. Colarelli, Team member selection decisions for virtual versus face-to-face teams. *Computers in Human Behavior* **26**, 4 (July 2010), 630–635.

[Guido *et al*., 2005] Guido Hertel, Susanne Geister and Udo Konradt, Managing virtual teams: A review of current empirical research. *Human Resource Management Review* **15**, 1 (March 2005), 69–95.

[Hambley *et al*., 2007] Laura A. Hambley, Thomas A. O'Neill and Theresa J.B. Kline, Virtual team leadership: the effects of leadership style and communication medium on team interaction styles and outcomes. *Organizational Behavior and Human Decision Processes* **103**, 1 (May 2007), 1–20.

[Kanawattanachai and Yoo, 2007] Prasert Kanawattanachai and Youngjin Yoo, The impact of knowledge coordination on virtual team performance over time. *MIS Quarterly* **31**, 4 (December 2007), 783–808.

[Larsen and McInerney, 2002] Kai R.T. Larsen and Claire R. McInerney, Preparing to work in the virtual organization. *Information & Management* **39**, 6 (May 2002), 445–456.

[Lee, 2008] Margaret R. Lee, E-ethical leadership for virtual project teams. *International Journal of Project Management* **27**, 5 (July 2009), 456–463.

[Lee-Kelley, 2006] Liz Lee-Kelley, Locus of control and attitudes to working in virtual teams. *International Journal of Project Management* **24**, 3 (April 2006), 234–243.

[Lurey and Raisinghani, 2001] Jeremy S. Lurey and Mahesh S. Raisinghani, An empirical study of best practices in virtual teams. *Information & Management* **38**, 8 (October 2001), 523–544.

[Nunamaker *et al*., 2009] Jay F. Jr Nunamaker, Bruce A. Reinig and Robert O. Briggs, Principles for effective virtual teamwork. *Communications of the ACM* **52**, 4 (April 2009), 113–117.

[Piccoli and Ives, 2003] Gabriele Piccoli and Blake Ives, Trust and the unintended effect of behavior control in virtual teams. MIS Quarterly **27**, 3 (September 2003), 365–395.

[Pokharel, 2011] Shaligram Pokharel, Stakeholders' roles in virtual project environment: a case study. *Journal of Engineering and Technology Management* **28**, 3 (July-September 2011), 201–214.

[Purvanova and Bono, 2009] Radostina K. Purvanova and Joyce E. Bono, Transformational leadership in context: face-to-face and virtual teams. *The leadership Quarterly* **20**, 3 (June 2009), 343–357.

[Reed and Knight, 2010] April H. Reed and Linda V. Knight, Effect of a virtual project team environment on communication-related project risk. *International Journal of Project Management* **28**, 5 (July 2010), 422–427.

[Schümmer and Lukosch, 2009] Till Schümmer and Stephan Lukosch, Understanding tools and practices for distributed pair programming. *Journal of Universal Computer Science* **15**, 16 (2009), 3101–3125.

Käytettävyys ja pelattavuus digitaalisten pelien tutkimuksessa

Anni Vuokko

Tiivistelmä.
Pelitutkimuksessa ja digitaalisten pelien käytettävyyttä tutkittaessa käsitteitä käytettävyys ja pelattavuus käytetään usein epäjohdonmukaisesti. Tässä tutkielmassa tarkoitukseni on määritellä termit pelitutkimuksen kontekstissa ja selventää niiden oikeaoppista käyttöä.

Avainsanat ja -sanonnat: Käytettävyys, pelattavuus
CR-luokat: C.2, C.3

1. Johdanto

Digitaalisten pelien tutkimus on suhteellisen uusi ja vähän tunnettu tieteenala. Kirjallisuutta on rajattu määrä, ja perusoppikirjoja aiheesta on vain muutama. Pelattavuus käsitteenä liittyy myös pelitutkimuksen perinteisempään haaraan eli pelin ja leikin tutkimukseen, mutta termi tässä kontekstissa eroaa hieman sen käytöstä digitaalisten pelien maailmassa.

Käytettävyys vuorovaikutteisen teknologian osana on pelattavuuteen verrattuna vanha käsite. Useat eri tutkijat ovat esittäneet omia määritelmiään käytettävyydelle, kirjallisuutta on runsaasti ja käytettävyydellä on jopa oma ISO-standardinsa. Sanan tarkoitus on laajasti tunnettu ja käytetty oikein ohjelmisto- ja käyttöliittymäsuunnittelun alalla.

Uusien tutkimusalojen terminologia on kuitenkin usein vakiintumatonta. Pelitutkimus on lisääntynyt 2000-luvulla konsolipelien ja PC-pelien yleistyessä tavallisissa kodeissa. Perinteisessä pelitutkimuksessa leikille ei välttämättä ole olemassa kohdetta, jonka pelattavuutta ja käytettävyyttä voitaisiin tutkia. Digitaalisten pelien tutkimus antaa kuitenkin oivat puitteet pelitarvikkeiden suunnittelun kehittämiselle.

Pelattavuus ja pelien käytettävyys ovat oleellisia elementtejä digitaalisten pelien tutkimuksessa. Tutkimus on kuitenkin monialaista, jolloin termistö on joskus sekavaa ja kahta käsittelemääni termiä käytetään harhaanjohtavasti erottelematta niitä selkeästi toisistaan. Tämän tutkielman tarkoitus on selventää

eroa käsitteiden käytettävyys ja pelattavuus välillä puhuttaessa digitaalisista peleistä.

Ensisijaisesti vuorovaikutteisen teknologian opiskelijana tulin ensimmäisellä pelitutkimuksen kurssillani miettineeksi pääeroja termien käytettävyys ja pelattavuus välillä. Ensimmäinen termi oli minulle tuttu opiskeluistani, mutta jälkimmäiseen tutustuin ensimmäistä kertaa mainitulla kurssilla. Kurssi oli erittäin keskustelupainotteinen, ja luennoilla pyrin tarkkailemaan sanojen käyttöä eri tilanteissa. Kurssin päätyttyä en vieläkään ollut ymmärtänyt kokonaan termien eroa, ja kandidaatin tutkielman aihetta pohtiessani tuli ongelma jälleen mieleeni. Suoritin pikaisen tiedonhaun aiheesta ja huomasin, että monessa lähteessä sanoja käytetään sekaisin ja lähes synonyymeina. Useissa tuloksissa molemmat sanat oli mainittu, mutta niitä ei ole erikoisemmin määritelty.

Pelattavuuden määrittelemättömyys ja käytettävyyden selvä sukulaisuus sanan kanssa sai minut kiinnostumaan aiheesta, ja tässä tutkielmassa toivon selvittäväni eroja näiden kahden termin välillä.

Tässä tutkielmassa käytettävyyttä tullaan tarkastelemaan digitaalisen käyttöliittymäsuunnittelun alalla. Tutkielma koskee käytettävyyttä ja pelattavuutta nimenomaan digitaalisessa kontekstissa.

2. Käytettävyys terminä

Hyvällä käytettävyydellä pyritään käyttäjän ja laitteen mielekkääseen yhteistoimintaan. Ohjelmien käyttäjien ei voida olettaa olevan aina kokeneita käyttäjiä, ja ohjelmat on siksi tehtävä helppokäyttöisiksi ja hyvin ymmärrettäviksi sekä tuotetta ennen käyttäneille että uusille käyttäjille. Käytettävyyttä tutkitaan muun muassa psykologian ja ihmisen ja koneen vuorovaikutuksen kautta. Yhteistoiminnasta pyritään tekemään mahdollisimman tehokasta ja miellyttävää [Sinkkonen et al., 2002].

Käytettävyys on mittari, joka tarkastelee asioiden helppokäyttöisyyttä. Tuotteen hyvä käytettävyys tarkoittaa, että käyttäjä pystyy käyttämään tuotetta tehokkaasti ja kokee käytön miellyttävänä. Käyttäjä myös löytää laitteesta haluamansa toiminnot nopeasti ja vaivattomasti. Käytettävyyden parantuessa myös järjestelmän luotettavuus paranee. Vuorovaikutuksen käyttäjän ja koneen välillä on oltava yksinkertaista ja luonnollista [Kuutti, 2003].

Käytettävyys koskee käyttäjän tavoitteita. Ennen kuin käyttäjä saavuttaa tavoitteensa, hän on asemassa jossa tyydyttävää tulosta ei vielä ole saavutettu. Tavoitteen ja nykytilan välillä on tietty välimatka, jonka hyvä käytettävyys pyr-

kii minimoimaan [Kurosu, 2007]. Tuotteen tulisi auttaa tehtävän suorittamisessa ja tehdä siitä helpompi kuin ilman tuotteen apua, sekä tukea niitä tehtäviä joiden suorittamiseen se on luotu.

ISO-standardin mukaan hyvä käytettävyys kattaa tehokkuuden, käytön miellyttävyyden sekä tuottavuuden. Ohjelmiston käytettävyyteen liittyy myös opittavuus, muistettavuus ja virheettömyys [Nielsen, 1993]. Opittavuus auttaa uutta käyttäjää käytön aloittamisessa sekä tuloksellisuudessa. Opittavuudelle on tärkeää ennustettavuus käyttäjän puolelta. Tällöin ohjelma toimii kuten sen oletetaan toimivan käyttäjän omien mentaalimallien ja aikaisemman ohjelmistokäytön perusteella.

Muistettavuudella tarkoitetaan käyttäjän kykyä palauttaa mieleen laitteen toiminnallisuus käytettyään sitä aikaisemmin. Tähän vaikuttaa oleellisesti se, miten helppoa käyttö on ollut edellisillä käyttökerroilla. Jos käyttäjä on silloin kokenut käytön hankalaksi, on todennäköistä että toiminnan mieleenpalauttaminen on hankalampaa kuin mieluisan käyttökerran jälkeen. Positiiviset kokemukset laitteesta tuovat käyttäjälle itsevarmuutta sen käyttämiseen [Kuutti, 2003].

Kuten ihmiselle on luonnollista, hyvää käytettävyyttä ajatellaan usein vasta sen puuttuessa. Hyvän käytettävyyden yksi ominaisuus onkin tavallaan näkymättömyys. Käytettävyyden ollessa erinomaisesti hoidettu, tuntuu tuotteen käyttö sujuvalta ja ongelmattomalta. Tällöin käyttäjä ei kiinnitä erityistä huomiota polkuun, jota kautta tiedonhaku tapahtuu, vaan voi keskittyä täydellisesti oman haasteeseensa ja ongelmansa ratkaisuun. Tätä käsittelen vielä myöhemmin luvussa 3.

Käytettävyyteen kiinnitetään nykyään enemmän huomiota sen merkityksen saatua rahallisen arvon. Esimerkiksi verkkokauppojen käyttäjät tilaavat tuotteensa mieluummin sivuilta, joiden käyttö on heille luonnollista ja miellyttävää. Käytettävyystestaus on lisääntynyt Internetin levitessä laajalle ja suurentaessa markkinoitaan verkkokauppojen ja mainonnan piireissä. Vasta tuotteen käyttöönoton ja myyntiin laittamisen jälkeen huomatut käytettävyysongelmat voivat tulla kalliiksi valmistajalle. Jo suunnittelun alkuvaiheissa huomattu ja korjattu ongelma ja jatkuva tuotteen käyttöliittymän käytettävyyden kehittäminen tuotetta luotaessa aiheuttaa vähemmän kustannuksia. Jatkuvalla kehityksellä estetään kriittiset käytettävyysongelmat tuotteessa sen päästessä kuluttajien saataville [Sinkkonen et al., 2002].

Jotta suunnittelijat osaisivat kehittää tuotteen sen kohderyhmälle sopivaksi, on heidän tehtävä yleistyksiä ryhmästä. Sinkkonen [2002] esittää, että ihmisen toiminnasta voidaan tietää joitakin melko universaaleja asioita. Toisaalta kaikki

säännöt tuntevat poikkeuksensa. Web-suunnittelussa on tavallista, etteivät suunnittelijat tiedä, keitä tuotteen käyttäjät tulevat olemaan, tai miten he haluavat toimia. Ihmisryhmien kulttuurierot tulee ottaa huomioon käytettävyystutkimuksessa. Kulttuuriin kuuluvat muun muassa puhekieli, asioiden merkitykset ja käsitteet. Vuorovaikutuksessa käyttäjän kanssa tulee käyttää hänen kieltänsä. Asiantuntijoiden termistö on usein peruskäyttäjälle vaikeasti ymmärrettävää. Tosin jos suunnittelutiimi tietää kenelle tuotetta ollaan suunnittelemassa, voidaan sanastoa muokata kohderyhmän mukaan. Jos suunnitellaan tuotetta työntekijöille yhtiössä, jossa palkolliset ovat esimerkiksi tietokonealan asiantuntijoita, voidaan ohjelmiston kielikin jättää asiantuntijamaisemmaksi. Käyttäjä voi turhautua liian vaikean kielen lisäksi liian yksinkertaisesta ilmaisusta ja liian vähästä teknisestä tiedosta esimerkiksi ongelmanratkaisutilainteissa.

Maailma jatkaa yhtenäistymistään kulttuurisesti, mutta on olemassa asioita, kuten värien symboliset arvot, joita voidaan vielä käyttää hyväksi tietylle ihmisryhmälle tuotteita suunnitellessa. Esimerkiksi länsimaissa punaisen, vihreän, mustan ja valkoisen värin merkityksiä pidetään laajasti sovittuina. Muilla alueilla merkitykset saattavat olla päinvastaisia, ja tuotteen värikoodausta täytyy silloin muuttaa käyttäjän mentaalimallien tapaamiseksi [Kuutti, 2003].

Käyttäjä kantaa itsessään psykologisia ja fysiologisia rakenteita sekä melko pysyviä kulttuurisia asioita, kuten normeja, tapoja ja kieltä. Ihmisen toimintaa rajoittavat ja laajentavat yksittäiset toimintamallit, kuten pelot tai erikoiset kyvyt. Tavalliset kyvyt ovat niitä kykyjä, joiden oletetaan kehittyvän jokaiselle terveenä syntyneelle ihmiselle, kuten näkö- ja muut aistit. Fysiologisiin rakenteisiin lasketaan myös muisti, jonka tutkimus on tärkeä osa käytettävyyden psykologiaa.

Terminä käytettävyydellä voidaan viitata sekä ongelmien vähentämiseen (non-negative usability) että ohjelman hyvin toimivien osien parantamiseen (positive usability). Molemmat kuitenkin pyrkivät samaan tavoitteeseen [Kurosu, 2007].

Digitaalisissa peleissä käytettävyydellä on suuri merkitys sekä yleisen pelikokemuksen että pelattavuuden kannalta.

3. Käytettävyys digitaalisten pelien tutkimuksessa

Digitaalisuus yleisesti on tuonut paljon uusia tutkimuskohteita käytettävyydelle. Pelien käytettävyyden arviointi suuntautuu pelin käyttöliittymään ja

kontrolleihin joita pelaaja käyttää ollessaan vuorovaikutuksessa pelin kanssa [Korhonen et al. 2009].

Hyvän pelikäyttöliittymän tavoite on, ettei käyttäjän tarvitsisi keskittyä varsinaiseen käyttöliittymään ollenkaan pyrkiessään toteuttamaan tavoitettaan. Tällöin käyttäjän koko aivokapasiteetti on peleissä varattu itse pelaamiselle ja sen haasteille, eikä käyttöliittymää ole tarpeen miettiä erikseen. Heuristiikat paremman käytettävyyden luomiseen ovat kehittyneet tekniikan kehittymisen ja pelitarjonnan kasvamisen mukana.

Se, mikä tekee pelin hyvästä käytettävyydestä erityisen tärkeää, on pelaamisen vapaaehtoisuus. Peliä ei pelata, jos sen yleisilme ja alkukosketukset ovat kömpelöitä ja vaikeita ymmärtää. Digitaalisten pelien pelaajien piirissä peliarvostelut sekä mielipiteet peleistä liikkuvat nopeasti niiden alustojen ansiosta. On pelin valmistajalle tärkeää, että pelaaja voi luottaa voivansa keskittyä pelin tarinaan ja haasteeseen ilman että hänen täytyy ponnistella jo itse käyttöliittymän kanssa. Pelimarkkinoilla riittää kilpailua, jolloin pelaajan ei tarvitse tyytyä tuotteeseen, vaan hän voi vapaasti valita muunlaisen pelin, muun peliyhtiön vastaavan pelin tai jopa vaihtaa kokonaan pelialustaa esimerkiksi PC:ltä konsoliin paremman kokemuksen saavuttamiseksi.

Yleinen käytettävyysongelma onkin juuri pelialustojen vaihtuvuus ja pelien pysyminen samanlaisina. Pelejä luodaan usein ensisijaisesti tietylle konsolille. Peliyhtiö voi kuitenkin haluta tuottaa peliään myös PC:lle, jonka ominaisuudet pelin ohjaukseen ovat konsolin kanssa täysin erilaiset. Kustannustehokkuuden varjolla muunnos pelialustalta toiselle saatetaan tehdä liian nopeasti ja huolimattomasti, jolloin laatu ja käytettävyys sekä pelikokemus kärsivät. Tämä saattaa vaikuttaa pelaajan päätökseen ostaa peli joko pelialustan perusteella tai yksinkertaisella päätöksellä jättää tuote kokonaan ostamatta.

Peliteollisuudessa pelisuunnittelijoiden on entistä vaikeampi ennustaa pelaajan tieto- ja taitotasoa peleissä. Nykyään pelejä pelaa suurempi kansanosa kuin ennen, jolloin pelikulttuuri oli vielä pienen vähemmistön intensiivinen harrastus. Tänä päivänä pelaajien ryhmä on ominaisuuksissaan ja pelihistoriassaan laajempi. Pelejä ei voida enää suunnitella pienelle asiantuntijaryhmälle.

Käytettävyys peleissä keskittyy pelien käyttöliittymään ja on osittain teknisempi aihe kuin pelattavuus. Käytettävyys on peliteollisuudessa toistaiseksi yllättävän pienessä osassa. Pelien käytettävyys näkyy peleissä valikoissa, alkunäytöissä sekä pelimekaniikkaan liittyvissä yksityiskohdissa. Valikot ovat suurin ja näkyvin haaste hyvän käytettävyyden kehittämisessä peleissä.

4. Pelattavuus terminä

Digitaalisten pelien tutkimus alkoi noin vuonna 2001, mutta pelitutkimus on paljon vanhempi käsite. Ennen digitaalisten pelien tutkimusta pelitutkimus tutki perinteisempiä pelejä ja leikkejä, kuten lasten kotileikkiä tai hippaa.

Pelitutkimus on hyvin monialaista, ja vain harva pelitutkimuksen otsikon alla esiintyvä tutkimus on pelialaa ensisijaisesti tutkivan tutkijan työtä – pelitutkimuksella itsellään ei ole vielä kovin kattavaa määrää tutkijoita, joten tutkimuksia aiheesta sekoitettuna omaan alaansa tekevät usein muiden alojen asiantuntijat. Näissä tutkielmissa pelitutkimus tosin on pienemmässä osassa tutkijan ensisijaisen tutkimusalueen ollessa vallitsevampi aihe. Digitaalisia pelejä voi tutkia esimerkiksi sosiologian, psykologian sekä kielitieteen näkökulmasta. Aloilla on tutkimukseen omat menetelmänsä, mikä tuo tutkimukseen laajuutta, mutta toisaalta vaikeuttaa kollegojen käyttämän termistön ymmärtämistä. Pelitutkimusta tehneistä tutkijoista voidaan siis erotella pelitutkijat, joiden alaa pelit ovat ensisijaisesti, ja pelejä tutkimuksissaan oman alansa kannalta tarkastelleet muiden alojen tutkijat, jotka myöskin asennoituvat tutkimuksen kohteisiin eri tavalla.

Myös kielimuuri saattaa vaikeuttaa termien täyttä ymmärtämistä. Suomen kielessä voimme erotella sanat ”peli” ja ”leikki”, mutta englanniksi joudumme tyytymään molemmissa tapauksissa sanaan ”game”. Tämä laajentaa sanan merkitystä ja sillä voidaan englannissa viitata koko pelitutkimuksen laajaan tutkimusalueeseen, kun taas suomen kielessä sanavalinta voi supistaa oletettua tutkimuksen alaa, vaikka suomenkin kielessä käytämme sanaa ”pelitutkija” molemmissa vaihtoehdoissa.

Pelattavuudelle, kuten käytettävyydellekin, on olemassa heuristiikkoja sen testaamiseen. HEP (The Heuristics Evaluation for Playability) on tiivistää 43 heuristiikkaa neljään kategoriaan: pelikokemus, pelin tarina, mekaniikka ja pelin käytettävyys. Isbister ja Schaffer [2008] esittelevät muutamia heuristiikkoja pelien ja mobiilipelien arvioimiseen ja kehuvat niiden laajuutta ja käytännöllisyyttä, mutta mainitsevat kuitenkin täydellisen listan puuttumisen toistaiseksi.

Pelattavuus terminä on sukua käytettävyydelle, jota taas on määritelty tehokkuuden, muistettavuuden ja opittavuuden kaltaisilla määritelmillä useiden ihmisen ja tietokoneen vuorovaikutuksen tutkijoiden toimesta. Pelattavuus nähdään usein käytettävyyden jatkona hauskanpidon teemassa, mutta joskus se määritellään koko pelikokemuksen ja pelimekaniikan tasoksi, mukaan laskettuna esimerkiksi grafiikoiden ja äänien taso ja immersion syvyys pelissä.

Eräs pelattavuuden malli jakaa sen neljään pääosaan: toiminnallinen pelattavuus, rakenteellinen pelattavuus, audiovisuaalinen pelattavuus sekä sosiaalinen pelattavuus. Toiminnallinen pelattavuus arvioi samoja asioita kuin pelin käytettävyys: ohjelmistoa ja kontrolleja. Rakenteellisessa pelattavuudessa on kyse pelin haasteiden ja sääntöjen tasosta ja nautinnollisuudesta. Audiovisuaalisessa pelattavuudessa nimensä mukaisesti käsitellään pelin grafiikoiden ja äänen vaikutusta pelikokemuksen tasoon. Sosiaalinen pelattavuus arvioi pelin sosiaalista puolta ja sen sopimista erilaisiin sosiaalisiin toimintoihin [Mäyrä, 2008].

Kasuaalipeleillä tarkoitetaan suhteellisen helppoja pelejä, jotka eivät ole yhtä laajoja kuin nykyaikaiset konsolipelit. Kasuaalipelit ovat tarjolla mahdollisimman laajalle yleisölle, ja ne ovat usein halpoja tai ilmaisia. Mobiilipelialustat ovat mahdollistaneet kasuaalipelien nopean leviämisen. Vaikka pelit perustuvatkin suurissa osin haasteisiin, on kasuaalipelien yleistyminen muokannut suurien peliyhtiöiden näkökantaa siihen, miten vaikea pelin tulee loppujen lopuksi pelaajalle olla. Valtavirran aatteena suunnittelijoilla on pelinkehitys, jossa suunnittelijat pyrkivät luomaan pelin, jossa pelaaja ymmärtää, miten hänen halutaan pelissä etenevän.

Pelattavuus on kuitenkin eri asia kuin pelikokemus. Pelattavuutta voi suunnitella, mutta pelikokemus on aina henkilökohtainen [Sinkkonen et al., 2002]. Terminä pelattavuutta on määritelty huomattavasti vähemmän kuin käytettävyyttä.

5. Pelattavuus osana käytettävyyttä

Eroa pelattavuudessa ja käytettävyydessä peleissä voidaan miettiä ajattelemalla helppoutta. On käytettävyyden kannalta tärkeää, että pelin valikot ovat helppokäyttöisiä ja selkeitä, jolloin pelaajan ei tarvitse kuluttaa aikaa valikoissa seikkailemiseen, kun peli itsessään houkuttaa jo. Pelattavuudessa helppous on eri roolissa – hyvää peliä luodessa ei pelistä tehdä tarkoituksellisen helppoa.

Pelaajan eri toimintojen on kuitenkin oltava helpoiksi tehtyjä. Jos pelaajan hahmo liikkuu kömpelösti ja peli on liikkumisen osalta huonosti kehitetty tai grafiikan puutteellisuus vaikeuttaa hahmon liikkumista selkeästi, on pelaajan oikeutettua väittää pelikokemuksen olevan huono tai vajavainen. First Person Shooter -peleissä pelimaailma esitetään pelaajan näkökulmasta ja pelin haaste on tuhota vihollisia tai muita kohteita erilaisilla aseilla pelistä riippuen. Esimerkiksi juuri näissä peleissä pelaajan on saatava vaihdettua erityisen valikon kautta asettaan nopeasti – jos vaihto on koodattu kömpelöksi, pelikokemus ja

pelattavuus kärsivät käyttäjän turhautuessa. Aseen vaihdon yhteydessä tapahtuvat ongelmat voivat aiheuttaa pelaajan tulevan itse ammutuksi, mikä ymmärrettävästi tekee kömpelöstä valikosta ongelmallisen pelaajan näkökulmasta.

Peliä ei tule kuitenkaan pelattavuuden nimissä tehdä helpoksi. Pelin genre on määrittävä tekijä siinä, mitkä asiat pelissä tehdään helpoiksi ja mille aiheille jätetään haastetta. Kaiken kaikkiaan pelit perustuvat haasteelle, ja haaste on se, miksi peliä pelataan. Siksi pelisuunnittelijan näkökulmasta kaikkea ei voi tehdä pelissä helpoksi. Pelattavuudessa haasteiden on oltava yhä haastavia, mutta pelaajalle on annettava välineet esteiden voittamiseksi. Nämä välineet on tehtävä miellyttäviksi käyttää, mutta annettava haasteen säilyä. Malonen [1982] mukaan pelin lopputuloksen on oltava epävarma, jolloin pelaaja tuntee olevansa aidosti vastuussa mahdollisista tapahtumista. Pelaajalle on annettava juuri oikea määrä haastetta hänen taitoihinsa perustuen.

Hyvä esimerkki tästä on Escape the Room -tyypin point-and-click -pelit, joissa pelaajan on paettava kolmiuloitteisesta suljetusta huoneesta käyttämällä huoneesta löytämiään esineitä, joita pelaaja voi poimia mukaansa ja mahdollisesti yhdistellä tai käyttää niitä yksinään poispääsyyn. Huone on interaktiivinen ja pelaaja löytää hiiren kursorin avulla tavaroita joihin hän voi koskea ja joita voi käyttää. Näissä peleissä pelin tekninen puoli on tärkeässä osassa käytettävyyttä ja pelattavuutta. Huoneessa navigoinnin ja esineiden poimimisen sekä pois laittamisen on oltava teknisesti onnistunutta, jotta pelaajan mietintätyö pelin haasteen suhteen ei katkea huonoon suunnitteluun. Pelaajaa ei saa jättää miettimään tekeekö hän jotain teknisesti väärin, vaan pelin palautteen on oltava selkeää.

Toinen ero käytettävyyden ja pelattavuuden välillä on niiden suhde immersioon. Immersio on uppoutumista peliin. Se on tila, jossa pelaaja unohtaa ympäristönsä ja itsensä uudessa, keinotekoisessa maailmassa. Immersiota tapahtuu esimerkiksi kirjaa lukiessa, elokuvia katsoessa tai tässä tapauksessa videopelejä pelatessa. SCI-mallissa kuvataan kolme eri immersion ryhmää: sensorinen, haasteellinen ja mielikuvituksellinen.

Sensorisessa immersiossa pelin maailma, yksityiskohtaiset kentät, kaunis grafiikka ja mukaansatempaavat äänet saavat pelaajan keskittymään pelimaailmaan enemmän kuin todellisuuteen. Haasteellinen immersio ilmenee, kun pelaajalle annetaan pelissä juuri oikea määrä haastetta hänen taitoihinsa nähden. Vaikeus voi olla motoriikassa tai ongelmanratkaisussa, tärkeintä on, että haasteen taso on pelaajalle juuri oikea. Mielikuvituksellisessa immersiossa pelaajalle tarjotaan pelin kautta mahdollisuus käyttää omaa mielikuvitustaan nauttiakseen pelin maailmasta, juonesta ja henkilöhahmoista. Pelaaja saattaa

tuntea kuuluvuutta maailmaan tai eläytyä tietyn hahmon mielenliikkeisiin ja tuntea hänen tunteensa [Ermi and Mäyrä, 2005]. Tässä korostuu pelin leikillinen puoli.

Käytettävyys ohjelmistoissa ja käyttöliittymissä ei pyri suoranaisesti immersioon, vaikka hyvä käytettävyys antaakin immersiolle paremmat lähtökohdat, kun käyttöliittymä ei häiritse pelaajaa, ja hän voi rauhassa keskittyä pelimaailmaan. Pelattavuuden kehityksessä peleissä taas pyritään tietoisesti takaamaan mahdollisuus immersiolle. Huomion arvoista on, ettei pelin hyvyys tai mielenkiintoisuus aina korreloi immersion tason kanssa – myös pelaajaa ärsyttävä peli voi olla immersoiva haasteellisen immersion tasolla, ja peli kasuaalipelien genressä voi olla hyvä olematta kovin immersoiva.

Myös haaste itsessään erottaa käytettävyyden pelattavuudesta. Pelattavuus pyrkii tekemään haasteen miellyttäväksi. Käytettävyys pyrkii minimoimaan ja lopulta eliminoimaan haasteen. Haasteen ratkaiseminen on pelattavuuden näkökulmasta leikki ja positiivinen asia.

On kiinnostavaa, että se, minkä laskemme nykyään hyväksi pelattavuudeksi, on kehittynyt osittain suurimpien pelien alkujaan tekemien suunnittelupäätösten perusteella. Ensimmäiset alaa villinneet pelit ovat kehittäneet pelaajille tietynlaisia mentaalimalleja esimerkiksi pelin kontrolleista, joita vastaan on vaikea taistella hyvän käytettävyyden nimissä. Kuten aiemmin mainittua, on käytettävyydessä tärkeää muistettavuus ja opittavuus, joihin vaikuttavat huomattavasti pelaajien mielleyhtymät siitä, millainen pelin ja sen käyttöliittymän pitäisi olla. On vaikeaa luoda täysin uudenlainen konsepti käyttöliittymään, sillä vaikka opittavuus olisi kunnossa, voivat käyttäjät kokea uuden tavan kaukaiseksi ja hylkiä uutta.

Käytettävyyden tapauksessa helppous ja helppokäyttöisyys toimivat lähes synonyymeinä, kun taas pelattavuus on enemmän miellyttävyyttä, sillä pelikokemukseen kuuluu myös haaste. Käytettävyys sisältää pelin mekaaniset osat, summaa ne pelidynamiikan kanssa ja muodostaa näiden aiheiden vuorovaikutteisuuden ja yhteistyön kautta pelattavuuden.

6. Yhteenveto

Pelien rakenteet luovat omat merkityksensä. Oikeassa elämässä monopolirahalla ei ole merkitystä, mutta pelin sisällä se on arvokasta valuuttaa [Costikyan, 2002]. Samoin pelitutkimus luo ja muokkaa omia käsitteitään ikääntyessään ja tullessaan tunnetummaksi.

Pelattavuus on vain yksi digitaalisen pelitutkimuksen vielä melko määrittelemättömistä termeistä. Sen suhde käytettävyyteen tekee siitä kuitenkin olennaisen, etenkin pelattavuuden ollessa tärkeä valtti peliteollisuudelle pelien myyntitilastoissa. Esimerkiksi Electronic Arts -pelituottajan julkaisemat NHL ja FIFA -urheilupelisarjojen pelien joka vuosi ilmestyvät uudet versiot myyvät aina hyvin muun muassa pienten pelattavuuteen tehtyjen parannusten ansiosta. Vannoutuneet peliharrastajat miettivät pelattavuutta ja keskustelevat asiasta – eivätkä he pelaa pelejä, jotka ärsyttävät heitä.

Digitaalisten pelien tutkimus on nuori tieteenala, jonka käsitteistö on vielä kehittymässä. Pelien ja etenkin digitaalisuuden ja pelillisyyden yleistyessä sekä lasten että aikuisten jokaisella elämänalueella on varmaa, että pelitutkimusta tullaan kehittämään. Myös käytettävyys tulee uusien tutkijoiden ansiosta saamaan tuoreita näkökulmia, laajennettuja määritelmiä ja uusia tutkimuksia muiden tieteenalojen tutkijoilta.

Vaikka käytettävyys ja pelattavuus voidaan erotella selkeästi, toimivat ne kuitenkin pelaajan mielessä pelissä käsi kädessä. Erot ovat selkeät, mutta yhteistyön on toimittava: hyvässä pelissä sekä pelattavuus että käytettävyys ovat kunnossa.

Viiteluettelo

[Costikyan, 2002] Greg Costikyan, I have no words and I must design, In: *Proceedings of Computer Games and Digital Cultures Conference.*

[Ermi and Mäyrä, 2005] Laura Ermi and Frans Mäyrä, Fundamental components of the gameplay experience: Analysing immmersion. In: Proc. DiGRA 2005 Conference: *Changing views – Worlds in Play*, 1–14.

[Isbister and Schaffer, 2008] Katherine Isbister and Noah Schaffer, *Game Usability*, Elsevier, 2008.

[Korhonen et al., 2009] Hannu Korhonen, Janne Paavilainen and Hannamari Saarenpää, Expert review method in game evaluations: comparison of two playability heuristic sets, In: *Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era*, 1–8.

[Kurosu, 2007] Masaaki Kurosu, Concept of usability revisited, In: *Proc. of HCI'07 Interaction Design and Usability*, 1–8.

[Kuutti, 2003] Wille Kuutti, *Käytettävyys, suunnittelu ja arviointi*. Talentum, 2003.

[Malone, 1982] Thomas W. Malone, Heuristics for designing enjoyable user interfaces: lessons from computer games. In: *Proc. of the 1982 Conference on Human Factors in Computing Systems (HCI'82)*, 63–68.

[Mäyrä, 2008] Frans Mäyrä, *Introduction to Game Studies: Games in Culture*. Sage Publications, 2008.

[Nielsen, 1993] Jacob Nielsen, *Usability Engineering*. Academic Press, 1993.

[Sinkkonen et al., 2002] Irmeli Sinkkonen, Hannu Kuoppala, Jarmo Parkkinen ja Raino Vastamäki, *Käytettävyyden psykologia.* Edita, 2002.